Efficient Computation of Representative Sets with Applications in Parameterized and Exact Algorithms[*]

FEDOR V. FOMIN[†]   DANIEL LOKSHTANOV[†]   SAKET SAURABH[†‡]

Abstract

Let $M = (E, \mathcal{I})$ be a matroid and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of subsets of E of size p. A subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is q-*representative* for $\mathcal{S}$ if for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. By the classical result of Bollobás, in a uniform matroid, every family of sets of size p has a q-representative family with at most $\binom{p+q}{p}$ sets. In his famous "two families theorem" from 1977, Lovász proved that the same bound also holds for any matroid representable over a field $\mathbb{F}$. As observed by Marx, Lovász's proof is constructive. In this paper we show how Lovász's proof can be turned into an algorithm constructing a q-representative family of size at most $\binom{p+q}{p}$ in time bounded by a polynomial in $\binom{p+q}{p}$, t, and the time required for field operations.

We demonstrate how the efficient construction of representative families can be a powerful tool for designing single-exponential parameterized and exact exponential time algorithms. The applications of our approach include the following.

- In the LONG DIRECTED CYCLE problem the input is a directed n-vertex graph G and the positive integer k. The task is to find a directed cycle of length at least k in G, if such a cycle exists. As a consequence of our $8^{k+o(k)}n^{\mathcal{O}(1)}$ time algorithm, we have that a directed cycle of length at least $\log n$, if such cycle exists, can be found in polynomial time. As it was shown by Björklund, Husfeldt, and Khanna [ICALP 2004], under an appropriate complexity assumption, it is impossible to improve this guarantee by more than a constant factor. Thus our algorithm not only improves over the best previous $\log n / \log \log n$ bound of Gabow and Nie [SODA 2004] but also closes the gap between known lower and upper bounds for this problem.

- In the MINIMUM EQUIVALENT GRAPH (MEG) problem we are seeking a spanning subdigraph D' of a given n-vertex digraph D with as few arcs as possible in which the reachability relation is the same as in the original digraph D. The existence of a single-exponential c^n-time algorithm for some constant $c > 1$ for MEG was open since the work of Moyles and Thompson [JACM 1969].

- To demonstrate the diversity of applications of the approach, we provide an alternative proof of the results recently obtained by Bodlaender, Cygan, Kratsch and Nederlof for algorithms on graphs of bounded treewidth, who showed that many "connectivity" problems such as HAMILTONIAN CYCLE or STEINER TREE can be solved in time $2^{\mathcal{O}(t)}n$ on n-vertex graphs of treewidth at most t. We believe that expressing graph problems in "matroid language" shed light on what makes it possible to solve connectivity problems single-exponential time parameterized by treewidth.

For the special case of uniform matroids on n elements, we give a faster algorithm computing a representative family in time $\mathcal{O}((\frac{p+q}{q})^q \cdot 2^{o(p+q)} \cdot t \cdot \log n)$. We use this algorithm

to provide the fastest known deterministic parameterized algorithms for k-PATH, k-TREE, and more generally, for k-SUBGRAPH ISOMORPHISM, where the k-vertex pattern graph is of constant treewidth. For example, our k-PATH algorithm runs in time $\mathcal{O}(2.851^k n \log^2 n)$.

Contents

1 Introduction

The theory of matroids provides a deep insight on the tractability of a number of fundamental problems in Combinatorial Optimizations like MINIMUM WEIGHT SPANNING TREE or PERFECT MATCHING. Marx in [33] was the first to apply matroids to design fixed-parameter tractable algorithms. The main tool used by Marx was the notion of *representative families*. Representative families for set systems were introduced by Monien in [35].

Let $M = (E, \mathcal{I})$ be a matroid and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of subsets of E of size p. A subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is q-*representative* for $\mathcal{S}$ if for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y and $\widehat{X} \cup Y \in \mathcal{I}$. In other words, if a set Y of size at most q can be extended to an independent set of size $|Y| + p$ by adding a subset from $\mathcal{S}$, then it also can be extended to an independent set of size $|Y| + p$ by adding a subset from $\widehat{\mathcal{S}}$ as well.

The Two-Families Theorem of Bollobás [8] for extremal set systems and its generalization to subspaces of a vector space of Lovász [29] (see also [19]) imply that every family of sets of size p has a q-representative family with at most $\binom{p+q}{p}$ sets. These theorems are the corner-stones in extremal set theory with numerous applications in graph and hypergraph theory, combinatorial geometry and theoretical computer science. We refer to Section 9.2.2 of [23], surveys of Tuza [42, 43], and Gil Kalai's blog[1] for more information on the theorems and their applications.

For set families, or equivalently for uniform matroids, Monien provided an algorithm computing a q-representative family of size at most $\sum_{i=0}^q p^i$ in time $\mathcal{O}(pq \cdot \sum_{i=0}^q p^i \cdot t)$ [35]. Marx in [32] provided another algorithm, also for uniform matroids, for finding q-representative families of size at most $\binom{p+q}{p}$ in time $\mathcal{O}(p^q \cdot t^2)$. For linear matroids, Marx [33] has shown how Lovász's proof can be transformed into an algorithm computing a q-representative family. However, the running time of the algorithm given in [33] is $f(p,q)(||A_M||t)^{\mathcal{O}(1)}$, where $f(p,q)$ is a polynomial in $(p+q)^p$ and $\binom{p+q}{p}$, that is, $f(p,q) = 2^{\mathcal{O}(p\log(p+q))} \cdot \binom{p+q}{p}^{\mathcal{O}(1)}$, and A_M is the matroid's representation matrix. Thus, when p is a constant, which is the way this lemma has been recently used in the kernelization algorithms [28], we have that $f(p,q) = (p+q)^{\mathcal{O}(1)}$. However, for unbounded p (for an example when $p = q = \frac{k}{2}$) the running time of this algorithm is bounded by $2^{\mathcal{O}(k\log k)}(||A_M||t)^{\mathcal{O}(1)}$.

Our results. We give two faster algorithms computing representative families and show how they can be used to obtain improved parameterized and exact exponential algorithms for several fundamental and well studied problems.

Our first result is the following

Theorem 1. *Let $M = (E, \mathcal{I})$ be a linear matroid of rank $p + q = k$ given together with its representation matrix A_M over a field $\mathbb{F}$. Let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of independent sets of size p. Then a q-representative family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ for $\mathcal{S}$ with at most $\binom{p+q}{p}$ sets can be found in $\mathcal{O}\left(\binom{p+q}{p}tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$ operations over $\mathbb{F}$. Here, $\omega < 2.373$ is the matrix multiplication exponent.*

Actually, we will prove a more general variant of Theorem 1 which allows sets to have weights. This extension will be used in several applications. This theorem uses the notion of weighted representative families and computes a weighted q-representative family of size at most $\binom{p+q}{p}$ within the running time claimed in Theorem 1. The proof of Theorem 1 relies on the exterior algebra based proof of Lovász [29] and exploits the multi-linearity of the determinant function.

For the case of uniform matroids, we provide the following theorem

Theorem 2. *Let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of sets of size p over a universe of size n. For a given q, a q-representative family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ for $\mathcal{S}$ with at most $\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ sets can be computed in time $\mathcal{O}((\frac{p+q}{q})^q \cdot 2^{o(p+q)} \cdot t \cdot \log n)$.*

The proof of Theorem 2 is essentially an algorithmic variant of the "random permutation" proof of Bollobás Lemma (see [23, Theorem 8.7]). A slightly weaker variant of Bollobás Lemma can be proved using random partitions instead of random permutations, the advantage of the random partitions proof being that it can be de-randomized using efficient constructions of *universal sets* [38]. To obtain our results we define *separating collections* and give efficient constructions of them.

Separating collections can be seen as a variant of universal sets. A *n-p-q-separating collection* $\mathcal{C}$ is a pair $(\mathcal{F}, \chi)$, where $\mathcal{F}$ is a family of sets over a universe U of size n and χ is a function

---

from $\binom{U}{p}$ to $2^{\mathcal{F}}$ such that the following two properties are satisfied; (a) for every $A \in \binom{U}{p}$ and $F \in \chi(A)$, $A \subseteq F$, (b) for every $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$, there is an $F \in \chi(A)$ such that $A \subseteq F$ and $F \cap B = \emptyset$. The *size* of $(\mathcal{F}, \chi)$ is $|\mathcal{F}|$, whereas the *max degree* of $(\mathcal{F}, \chi)$ is $\max_{A \in \binom{U}{p}} |\chi(A)|$.

Here 2^S for a set S is the family of all subsets of S while $\binom{S}{p}$ is the family of all subsets of S of size p.

An efficient construction of separating collections is an algorithm that given n, p and q outputs the family $\mathcal{F}$ of a separating collection $(\mathcal{F}, \chi)$ and then allows queries $\chi(A)$ for $A \in \binom{U}{p}$. We give constructions of separating collections of optimal (up to subexponential factors in $p+q$) size and degree, and construction and query time which is linear (up to subexponential factors in $p + q$) in the size of the output.

Applications. Here we provide the list of main applications that can be derived from our algorithms that compute representative families together with a short overview of previous work on each application.

Reference	Randomized	Deterministic
Monien [35]	-	$\mathcal{O}(k!nm)$
Bodlaender [6]	-	$\mathcal{O}(k!2^k n)$
Alon et al. [1]	$\mathcal{O}(5.44^k n)$	$\mathcal{O}(c^k n \log n)$ for a large c
Kneis at al. [25]	$\mathcal{O}^*(4^k)$	$\mathcal{O}^*(16^k)$
Chen et al. [10]	$\mathcal{O}(4^k k^{2.7} m)$	$4^{k+\mathcal{O}(\log^3 k)} nm$
Koutis [26]	$\mathcal{O}^*(2.83^k)$	-
Williams [44]	$\mathcal{O}^*(2^k)$	-
Björklund et al. [4]	$\mathcal{O}^*(1.66^k)$	-
This paper	-	$\mathcal{O}(2.851^k n \log^2 n)$

Table 1: Results for k-PATH. We use $\mathcal{O}^*()$ notation that hides factors polynomial in the number of vertices n and the parameter k in cases when the authors do not specify the power of polynomials.

k-**Path.** In the k-PATH problem we are given an undirected n-vertex graph G and integer k. The question is if G contains a path of length k. k-PATH was studied intensively within the parameterized complexity paradigm [15]. For n-vertex graphs the problem is trivially solvable in time $\mathcal{O}(n^k)$. Monien [35] and Bodlaender showed that the problem is fixed parameter tractable. Monien used representative families for set systems for his k-PATH algorithm [35] and Plehn and Voigt extended this algorithm to SUBGRAPH ISOMORPHISM in [41]. This led Papadimitriou and Yannakakis [40] to conjecture that the problem is solvable in polynomial time for $k = \log n$. This conjecture was resolved in a seminal paper of Alon et al. [1], who introduced the method of color-coding and obtained the first single exponential algorithm for the problem. Actually, the method of Alon et al. can be applied for more general problems, like finding a k-path in directed graphs, or to solve the SUBGRAPH ISOMORPHISM problem in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(t)}$, when the treewidth of the pattern graph is bounded by t. There has been a lot of efforts in parameterized algorithms to reduce the base of the exponent of both deterministic as well as the randomized algorithms for the k-PATH problem, see Table 1. After the work of Alon et al. [1], there were several breakthrough ideas leading to faster and faster *randomized* algorithms. Concerning deterministic algorithms, no improvements occurred since 2007, when Chen et al. [11] showed a clever way of applying universal sets to reduce the running time of color-coding algorithm to $\mathcal{O}^*(4^{k+o(k)})$.

k-PATH is a special case of the k-SUBGRAPH ISOMORPHISM problem, where for given n-vertex graph G and k-vertex graph F, the question is whether G contains a subgraph isomorphic to F. In addition to k-PATH, parameterized algorithms for two other variants of k-SUBGRAPH ISOMORPHISM, when F is a tree, and more generally, a graph of treewidth at most t, were studied in the literature. Alon et al. [1] showed that k-SUBGRAPH ISOMORPHISM, when the treewidth of the pattern graph is bounded by t, is solvable in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(t)}$. Cohen et al. gave a randomized algorithm that for an input digraph D decides in time $5.704^k n^{\mathcal{O}(1)}$ if D contains a given out-tree with k vertices [12]. They also showed how to derandomize the algorithm in time $6.14^k n^{\mathcal{O}(1)}$. Amini et al. [2] introduced an inclusion-exclusion based approach in the classical color-coding and gave a randomized $5.4^k n^{\mathcal{O}(t)}$ time algorithm and a deterministic $5.4^{k+o(k)}n^{\mathcal{O}(t)}$ time algorithm for the case when F has treewidth at most t. Koutis and Williams [27] generalized their algebraic approach for k-PATH to k-TREE and obtained a randomized algorithm running in time $2^k n^{\mathcal{O}(1)}$ for k-TREE. A superset of the authors in [18], extended this result by providing a randomized algorithm for k-SUBGRAPH ISOMORPHISM running in time $2^k (nt)^{\mathcal{O}(t)}$, when the treewidth of F is at most t. However, the fastest known deterministic algorithm for this problem prior to this paper, was the time $5.4^{k+o(k)}n^{\mathcal{O}(t)}$ algorithm from [2]. In this paper we give deterministic algorithms for k-PATH and k-TREE that run in time $\mathcal{O}(2.851^k n \log^2 n)$ and $\mathcal{O}(2.851^k n^{\mathcal{O}(1)})$. The algorithm for k-TREE can be generalized to k-SUBGRAPH ISOMORPHISM for the case when the pattern graph F has treewidth at most t. This algorithm will run in time $\mathcal{O}(2.851^k n^{\mathcal{O}(t)})$. Our approach can also be applied to find directed paths and cycles of length k in time $\mathcal{O}(2.851^k m \log^2 n)$ and $\mathcal{O}(2.851^k n^{\mathcal{O}(1)})$ respectively.

Longest Directed Cycle. In the LONGEST DIRECTED CYCLE problem we are interested in finding a cycle of length at least k in a directed graph. For this problem we give an algorithm of running time $\mathcal{O}(8^{k+o(k)}mn^2 \log n)$.

While at the first glance the problem is similar to the problem of finding a cycle or a path of length exactly k, it is more tricky. The reason is that the problem of finding a cycle of length $\geq k$ may entail finding a much longer, potentially even a Hamiltonian cycle. This is why color-coding, and other techniques applicable to k-PATH do not seem work here. Even for undirected graphs color-coding alone is not sufficient, and one needs an additional clever trick trick to make it work. The first fixed-parameter tractable algorithm for LONGEST DIRECTED CYCLE is due to Gabow and Nie [20], who gave algorithms with expected running time $k^{2k}2^{\mathcal{O}(k)}nm$ and worst-case times $\mathcal{O}(k^{2k}2^{\mathcal{O}(k)}nm \log n)$ or $\mathcal{O}(k^{3k}nm)$. These running times allow them to find a directed cycle of length at least $\log n / \log \log n$ in polynomial time. Let us note, that our algorithm implies that one can find in polynomial time a directed cycle of length at least $\log n$ if there is such a cycle. On the other hand, Björklund et al. [5] have shown that assuming Exponential Time Hypothesis (ETH) of Impagliazzo et al. [22], there is no polynomial time algorithm that finds a directed cycle of length $\Omega(f(n)\log n)$, for any nondecreasing, unbounded, polynomial time computable function f that tends to infinity. Thus, our work closes the gap between the upper and lower bounds for this problem.

Minimum Equivalent Graph. Our next application is from exact exponential time algorithms, we refer to [17] for an introduction to the area of exact algorithms. In the MINIMUM EQUIVALENT GRAPH (MEG) problem we are seeking a spanning subdigraph D' of a given digraph D with as few arcs as possible in which the reachability relation is the same as in the original digraph D. In other words, for every pair of vertices u, v, there is a path from u to v in D' if and only if the original digraph D has such a path. We show that this problem is solvable in time $\mathcal{O}(2^{4\omega n}m^2 n)$, where n is the number of vertices and m is the number of arcs in D.

MEG is a classical NP-hard problem generalizing the HAMILTONIAN CYCLE problem, see Chapter 12 of the book [3] for an overview of combinatorial and algorithmic results on MEG.

The algorithmic studies of MEG can be traced to the work of Moyles and Thompson [36] from 1969, who gave a (non-trivial) branching algorithm solving MEG in time $\mathcal{O}(n!)$. In 1975, Hsu in [21] discovered a mistake in the algorithm of Moyles and Thompson, and designed a different branching algorithm for this problem. Martello [30] and Martello and Toth [31] gave another branching based algorithm with running time $\mathcal{O}(2^m)$. No single-exponential exact algorithm, i.e. of running time $2^{\mathcal{O}(n)}$, for MEG was known prior to our work.

As it was already observed by Moyles and Thompson [36] the hardest instances of MEG are strong digraphs. A digraph is strong if for every pair of vertices $u \neq v$, there are directed paths from u to v and from v to u. MEG restricted to strong digraphs is known as the MINIMUM SCSS (strongly connected spanning subgraph) problem. It is known that the MEG problem reduces in linear time to MINIMUM SCSS, see e.g. [13].

Treewidth algorithms. We show that efficient computation of representative families can be used to obtain in time $2^{\mathcal{O}(t)}n$, where t is the treewidth of the input n-vertex graph, algorithms solving "connectivity" problems like HAMILTONIAN CYCLE or STEINER TREE. It is well known that many intractable problems can be solved efficiently when the input graph has bounded treewidth. Moreover, many fundamental problems like MAXIMUM INDEPENDENT SET or MINIMUM DOMINATING SET can be solved in time $2^{\mathcal{O}(t)}n$, where t is the treewidth of the input n-vertex graph. On the other hand, it was believed until very recently that for some "connectivity" problems such as HAMILTONIAN CYCLE or STEINER TREE no such algorithm exists. In their breakthrough paper, Cygan et al. [14] introduced a new algorithmic framework called Cut&Count and used it to obtain $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$ time Monte Carlo algorithms for a number of connectivity problems. Very recently, Bodlaender et al. [7] obtained the first deterministic single exponential algorithms for these problems. Bodlaender et al. presented two approaches, one based on rank estimations in specific matrices and the second based on matrix-tree theorem and computation of determinants. Our approach, based on representative families in matroids, can be seen as an alternative path to obtaining similar results. The main idea behind our approach is that all the relevant information about "partial solutions" in bags of the tree decomposition, can be encoded as an independent set of a specific matroid. Here efficient computation of representative families comes into play.

In all our applications we first define a specific matroid and then show a combinatorial relation between solution to the problem and independent sets of the matroid. Then we compute representative families using Theorem 1 or Theorem 2 and use them to obtain a solution to the problem. We believe that expressing graph problems in "matroid language" is a generic technique explaining why certain problems admit single-exponential parameterized and exact exponential algorithms.

Organization of the paper. In Section 2 we give the necessary definitions and state some of the known results that we will use. In Section 3 we prove Theorem 1 by giving an efficient algorithm for the computation of representative families for linear matroids. In Section 4 we prove Theorem 2 by giving an efficient algorithm for the computation of representative families for uniform matroids. In Section 5 we give all our applications of Theorems 1 and 2. Concluding remarks can be found in Section 6. The proofs of Theorem 1 and Theorem 2 are independent of each other and may be read independently. All of our applications use Theorems 1 and 2 as black boxes, and thus may be read independently of the sections describing the efficient computation of representative families.

2 Preliminaries

In this section we give various definitions which we make use of in the paper.

Graphs. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. A graph G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The subgraph G' is called an *induced subgraph* of G if $E(G') = \{uv \in E(G) \mid u, v \in V(G')\}$, in this case, G' is also called the subgraph *induced by* $V(G')$ and denoted by $G[V(G')]$. For a vertex set S, by $G \setminus S$ we denote $G[V(G) \setminus S]$. By $N(u)$ we denote (open) neighborhood of u, that is, the set of all vertices adjacent to u. Similarly, by $N[u] = N(u) \cup \{u\}$ we define the closed neighborhood. The degree of a vertex v in G is $|N_G(v)|$ and is denoted by $d(v)$. For a subset $S \subseteq V(G)$, we define $N[S] = \cup_{v \in S} N[v]$ and $N(S) = N[S] \setminus S$. By the length of the path we mean the number of edges in it.

Digraphs. Let D be a digraph. By $V(D)$ and $A(D)$ we represent the vertex set and arc set of D, respectively. Given a subset $V' \subseteq V(D)$ of a digraph D, let $D[V']$ denote the digraph induced by V'. A digraph D is *strong* if for every pair x, y of vertices there are directed paths from x to y and from y to x. A maximal strongly connected subdigraph of D is called a *strong component*. A vertex u of D is an *in-neighbor* (*out-neighbor*) of a vertex v if $uv \in A(D)$ ($vu \in A(D)$, respectively). The *in-degree* $d^-(v)$ (*out-degree* $d^+(v)$) of a vertex v is the number of its in-neighbors (out-neighbors). We denote the set of in-neighbors and out-neighbors of a vertex v by $N^-(v)$ and $N^+(v)$ correspondingly. A *closed directed walk* in a digraph D is a sequence $v_0 v_1 \cdots v_\ell$ of vertices of D, not necessarily distinct, such that $v_0 = v_\ell$ and for every $0 \leq i \leq \ell - 1$, $v_i v_{i+1} \in A(D)$.

Sets, Functions and Constants. Let $[n] = \{1, \ldots, n\}$ and $\binom{[n]}{i} = \{X \mid X \subseteq [n], |X| = i\}$.

Definition 2.1. *Given two families of sets $\mathcal{L}_1$ and $\mathcal{L}_2$, we define*

$$\mathcal{L}_1 \bullet \mathcal{L}_2 = \{X \cup Y \mid X \in \mathcal{L}_1 \text{ and } Y \in \mathcal{L}_2 \text{ and } X \cap Y = \emptyset\}.$$

Let $\mathcal{L}_1, \ldots, \mathcal{L}_r$ be r families. Then

$$\prod_{i \in [r]}^{\bullet} \mathcal{L}_i = \mathcal{L}_1 \bullet \cdots \bullet \mathcal{L}_r.$$

Definition 2.2. *For two families $\mathcal{A}$ and $\mathcal{B}$ over (subsets of) U, we define*

$$\mathcal{A} \circ \mathcal{B} = \{A \cup B \ : \ A \in \mathcal{A} \wedge B \in \mathcal{B}\}.$$

Definition 2.3. *For a family $\mathcal{A}$ over (subsets of) U and set X, we define*

$$\mathcal{A} \oplus X = \{A \cup X \ : \ A \in \mathcal{A}\}.$$

Throughout the paper we use ω to denote the matrix multiplication exponent. The current best known bound on $\omega < 2.373$ [45]. We use e to denote the base of natural logarithm.

2.1 Randomized Algorithms

We follow the same notion of randomized algorithms as described in [33, Section 2.3]. That is, some of the algorithms presented in this paper are randomized, which means that they can produce incorrect answer, but the probability of doing so is small. We assume that the algorithm has an integer parameter P given in unary, and the probability of incorrect answer is 2^{-P}.

2.2 Matroids

In the next few subsections we give definitions related to matroids. For a broader overview on matroids we refer to [39].

Definition 2.4. *A pair* $M = (E, \mathcal{I})$, *where* E *is a ground set and* $\mathcal{I}$ *is a family of subsets (called independent sets) of* E, *is a* matroid *if it satisfies the following conditions:*

(I1) $\phi \in \mathcal{I}$.

(I2) *If* $A' \subseteq A$ *and* $A \in \mathcal{I}$ *then* $A' \in \mathcal{I}$.

(I3) *If* $A, B \in \mathcal{I}$ *and* $|A| < |B|$, *then* $\exists\, e \in (B \setminus A)$ *such that* $A \cup \{e\} \in \mathcal{I}$.

The axiom (I2) is also called the hereditary property and a pair $(E, \mathcal{I})$ satisfying only (I2) is called hereditary family. An inclusion wise maximal set of $\mathcal{I}$ is called a *basis* of the matroid. Using axiom (I3) it is easy to show that all the bases of a matroid have the same size. This size is called the *rank* of the matroid M, and is denoted by $\mathsf{rank}(M)$.

2.3 Linear Matroids and Representable Matroids

Let A be a matrix over an arbitrary field $\mathbb{F}$ and let E be the set of columns of A. Given A we define the matroid $M = (E, \mathcal{I})$ as follows. A set $X \subseteq E$ is independent (that is $X \in \mathcal{I}$) if the corresponding columns are linearly independent over $\mathbb{F}$. The matroids that can be defined by such a construction are called *linear matroids*, and if a matroid can be defined by a matrix A over a field $\mathbb{F}$, then we say that the matroid is representable over $\mathbb{F}$. That is, a matroid $M = (E, \mathcal{I})$ is representable over a field $\mathbb{F}$ if there exist vectors in $\mathbb{F}^d$ that correspond to the elements such that the linearly independent sets of vectors precisely correspond to independent sets of the matroid. Here, $d = \mathsf{rank}(M)$. A matroid $M = (E, \mathcal{I})$ is called *representable* or *linear* if it is representable over some field $\mathbb{F}$.

2.4 Direct Sum of Matroids.

Let $M_1 = (E_1, \mathcal{I}_1)$, $M_2 = (E_2, \mathcal{I}_2)$, ..., $M_t = (E_t, \mathcal{I}_t)$ be t matroids with $E_i \cap E_j = \emptyset$ for all $1 \leq i \neq j \leq t$. The direct sum $M_1 \oplus \cdots \oplus M_t$ is a matroid $M = (E, \mathcal{I})$ with $E := \bigcup_{i=1}^{t} E_i$ and $X \subseteq E$ is independent if and only if for all $i \leq t$, $X \cap E_i \in \mathcal{I}_i$. Let A_i be the representation matrix of $M_i = (E_i, \mathcal{I}_i)$. Then,

$$
A_M = \begin{pmatrix}
A_1 & 0 & 0 & \cdots & 0 \\
0 & A_2 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & A_t
\end{pmatrix}
$$

is a representation matrix of $M_1 \oplus \cdots \oplus M_t$. The correctness of this construction is proved in [33].

Proposition 2.1 ([33, Proposition 3.4])**.** *Given representations of matroids* $M_1, \ldots, M_t$ *over the same field* $\mathbb{F}$, *a representation of their direct sum can be found in polynomial time.*

2.5   Uniform and Partition Matroids

A pair $M = (E, \mathcal{I})$ over an n-element ground set E, is called a uniform matroid if the family of independent sets is given by $\mathcal{I} = \{A \subseteq E \mid |A| \leq k\}$, where k is some constant. This matroid is also denoted as $U_{n,k}$. Every uniform matroid is linear and can be represented over a finite field by a $k \times n$ matrix A_M where the $A_M[i,j] = j^{i-1}$.

$$A_M = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ 1 & 2^2 & 3^2 & \cdots & n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{k-1} & 3^{k-1} & \cdots & n^{k-1} \end{pmatrix}$$

Observe that for A_M to be representable over a finite field $\mathbb{F}$, we need that the determinant of any $k \times k$ submatrix of A_M must not vanish over $\mathbb{F}$. The determinant of any $k \times k$ submatrix of A_M is upper bounded by $k! \times n^{k^2}$ (this follows from the Laplace expansion of determinants). Thus, choosing a field $\mathbb{F}$ of size larger than $k! \times n^{k^2}$ suffices.

A partition matroid $M = (E, \mathcal{I})$ is defined by a ground set E being partitioned into (disjoint) sets $E_1, \ldots, E_\ell$ and by ℓ non-negative integers $k_1, \ldots, k_\ell$. A set $X \subseteq E$ is independent if and only if $|X \cap E_i| \leq k_i$ for all $i \in \{1, \ldots, \ell\}$. Observe that a partition matroid is a direct sum of uniform matroids $U_{|E_1|, k_1}, \cdots, U_{|E_\ell|, k_\ell}$. Thus, by Proposition 2.1 and the fact that a uniform matroid $U_{n,k}$ is representable over a field $\mathbb{F}$ of size larger than $k! \times n^{k^2}$ we have that.

Proposition 2.2 ([33, Proposition 3.5])**.** *A representation over a field of size $\mathcal{O}(k! \times |E|^{k^2})$ of a partition matroid can be constructed in polynomial time.*

2.6   Graphic Matroids

Given a graph G, a graphic matroid $M = (E, \mathcal{I})$ is defined by taking elements as edge of G (that is $E = E(G)$) and $F \subseteq E(G)$ is in $\mathcal{I}$ if it forms a spanning forest in the graph G. The graphic matroid is representable over any field of size at least 2. Consider the matrix A_M with a row for each vertex $i \in V(G)$ and a column for each edge $e = ij \in E(G)$. In the column corresponding to $e = ij$, all entries are 0, except for a 1 in i or j (arbitrarily) and a -1 in the other. This is a representation over reals. To obtain a representation over a field $\mathbb{F}$, one simply needs to take the representation given above over reals and simply replace all -1 by the additive inverse of 1

Proposition 2.3 ([39])**.** *Graphic matroids are representable over any field of size at least 2.*

2.7   Truncation of a Matroid.

The *t-truncation* of a matroid $M = (E, \mathcal{I})$ is a matroid $M' = (E, \mathcal{I}')$ such that $S \subseteq E$ is independent in M' if and only if $|S| \leq t$ and S is independent in M (that is $S \in \mathcal{I}$).

Proposition 2.4 ([33, Proposition 3.7])**.** *Given a matroid M with a representation A over a finite field $\mathbb{F}$ and an integer t, a representation of the t-truncation M' can be found in randomized polynomial time.*

3   Fast Computation for Representative Sets for Linear Matroids

In this section we give an algorithm to find a q-representative family of a given family. We start with the definition of a *q-representative family*.

Definition 3.1 (*q*-**Representative Family**). *Given a matroid $M = (E, \mathcal{I})$ and a family $\mathcal{S}$ of subsets of E, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is q-representative for $\mathcal{S}$ if the following holds: for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. If $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is q-representative for $\mathcal{S}$ we write $\widehat{\mathcal{S}} \subseteq_{rep}^{q} \mathcal{S}$.*

In other words if some independent set in $\mathcal{S}$ can be extended to a larger independent set by q new elements, then there is a set in $\widehat{\mathcal{S}}$ that can be extended by the same q elements. A weighted variant of q-representative families is defined as follows. It is useful for solving problems where we are looking for objects of maximum or minimum weight.

Definition 3.2 (**Min/Max** *q*-**Representative Family**). *Given a matroid $M = (E, \mathcal{I})$, a family $\mathcal{S}$ of subsets of E and a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$ we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is* min q-representative *(*max q-representative*) for $\mathcal{S}$ if the following holds: for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with*

1. *$\widehat{X} \cup Y \in \mathcal{I}$; and*

2. *$w(\widehat{X}) \leq w(X)$ $(w(\widehat{X}) \geq w(X))$.*

We use $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ $(\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S})$ to denote a min q-representative (max q-representative) family for $\mathcal{S}$.

We say that a family $\mathcal{S} = \{S_1, \ldots, S_t\}$ of independent sets is a *p-family* if each set in $\mathcal{S}$ is of size p.

We start by three lemmata providing basic results about representative sets. These lemmata will be used in Section 5, where we provide algorithmic applications of representative families. We proof them for unweighted representative families but they can be easily modified to work for weighted variant.

Lemma 3.1. *Let $M = (E, \mathcal{I})$ be a matroid and $\mathcal{S}$ be a family of subsets of E. If $\mathcal{S}' \subseteq_{rep}^{q} \mathcal{S}$ and $\widehat{\mathcal{S}} \subseteq_{rep}^{q} \mathcal{S}'$, then $\widehat{\mathcal{S}} \subseteq_{rep}^{q} \mathcal{S}$.*

Proof. Let $Y \subseteq E$ of size at most q such that there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$. By the definition of q-representative family we have that there is a set $X' \in \mathcal{S}'$ disjoint from Y with $X' \cup Y \in \mathcal{I}$. Now the fact that $\widehat{\mathcal{S}} \subseteq_{rep}^{q} \mathcal{S}'$ yields that there exists a $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. □

Lemma 3.2. *Let $M = (E, \mathcal{I})$ be a matroid and $\mathcal{S}$ be a family of subsets of E. If $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_\ell$ and $\widehat{\mathcal{S}}_i \subseteq_{rep}^{q} \mathcal{S}_i$, then $\cup_{i=1}^{\ell} \widehat{\mathcal{S}}_i \subseteq_{rep}^{q} \mathcal{S}$.*

Proof. Let $Y \subseteq E$ of size at most q such that there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$. Since $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_\ell$, there exists an i such that $X \in \mathcal{S}_i$. This implies that there exists a $\widehat{X} \in \widehat{\mathcal{S}}_i \subseteq \cup_{i=1}^{\ell} \widehat{\mathcal{S}}_i$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. □

Lemma 3.3. *Let $M = (E, \mathcal{I})$ be a matroid of rank k and $\mathcal{S}_1$ be a p_1-family of independent sets, $\mathcal{S}_2$ be a p_2-family of independent sets, $\widehat{\mathcal{S}}_1 \subseteq_{rep}^{k-p_1} \mathcal{S}_1$ and $\widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_2} \mathcal{S}_2$. Then $\widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_1-p_2} \mathcal{S}_1 \bullet \mathcal{S}_2$.*

Proof. Let $Y \subseteq E$ of size at most $q = k - p_1 - p_2$ such that there is a set $X \in \mathcal{S}_1 \bullet \mathcal{S}_2$ disjoint from Y with $X \cup Y \in \mathcal{I}$. This implies that there exist $X_1 \in \mathcal{S}_1$ and $X_2 \in \mathcal{S}_2$ such that $X_1 \cup X_2 = X$ and $X_1 \cap X_2 = \emptyset$. Since $\widehat{\mathcal{S}}_1 \subseteq_{rep}^{k-p_1} \mathcal{S}_1$, we have that there exists a $\widehat{X}_1 \in \widehat{\mathcal{S}}_1$ such

that $\widehat{X}_1 \cup X_2 \cup Y \in \mathcal{I}$ and $\widehat{X}_1 \cap (X_2 \cup Y) = \emptyset$. Now since $\widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_2} \mathcal{S}_2$, we have that there exists a $\widehat{X}_2 \in \widehat{\mathcal{S}}_2$ such that $\widehat{X}_1 \cup \widehat{X}_2 \cup Y \in \mathcal{I}$ and $\widehat{X}_2 \cap (\widehat{X}_1 \cup Y) = \emptyset$. This shows that $\widehat{X}_1 \cup \widehat{X}_2 \in \widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2$ and $\widehat{X}_1 \cup \widehat{X}_2 \cup Y \in \mathcal{I}$ thus $\widehat{\mathcal{S}}_1 \bullet \widehat{\mathcal{S}}_2 \subseteq_{rep}^{k-p_1-p_2} \mathcal{S}_1 \bullet \mathcal{S}_2$. $\qquad\square$

The main result of this section is that given a representable matroid $M = (E, \mathcal{I})$ of rank $k = p+q$ with its representation matrix A_M and a p-family of independent sets $\mathcal{S} = \{S_1, \ldots, S_t\}$, a non-negative weight function $w : \mathcal{S} \to \mathbb{N}$, we can compute $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ and $\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$ of size $\binom{p+q}{p}$ deterministically in time $\mathcal{O}\left(\binom{p+q}{p} t p^{\omega} + t \binom{p+q}{q}^{\omega-1}\right)$. The proof for this result is obtained by making the known exterior algebra based proof of Lovász [29, Theorem 4.8] algorithmic. Although our proof is based on exterior algebra and is essentially the same as the proof given in [29], we give a proof here which avoids the terminology from exterior algebra and argues everything using the properties of determinants, thus making the proof self-contained.

For our proof we also need the following well-known generalized Laplace expansion of determinants. For a matrix $A = (a_{ij})$, the row set and the column set are denoted by $\mathbf{R}(A)$ and $\mathbf{C}(A)$ respectively. For $I \subseteq \mathbf{R}(A)$ and $J \subseteq \mathbf{C}(A)$, $A[I, J] = (a_{ij} \mid i \in I, j \in J)$ means the submatrix (or minor) of A with the row set I and the column set J. For $I \subseteq [n]$ let $\bar{I} = [n] \setminus I$ and $\sum I = \sum_{i \in I} i$.

Proposition 3.1 (Generalized Laplace expansion)**.** *For an $n \times n$ matrix A and $J \subseteq \mathbf{C}(A) = [n]$, it holds that*

$$\det(A) = \sum_{I \subseteq [n], |I|=|J|} (-1)^{\sum I + \sum J} \det(A[I, J]]) \det(A[\bar{I}, \bar{J}])$$

We refer to [37, Proposition 2.1.3] for a proof of the above identity. We always assume that the number of rows in the representation matrix A_M of M over a field $\mathbb{F}$ is equal to $\mathsf{rank}(M) = \mathsf{rank}(A_M)$. Otherwise, using Gaussian elimination we can obtain a matrix of the desired kind in polynomial time. See [33, Proposition 3.1] for details. We do not give the proof for Theorem 1 but rather for the following generalization.

Theorem 3. *Let $M = (E, \mathcal{I})$ be a linear matroid of rank $p + q = k$, $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets and $w : \mathcal{S} \to \mathbb{N}$ be a non-negative weight function. Then there exists $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$) of size $\binom{p+q}{p}$. Moreover, given a representation A_M of M over a field $\mathbb{F}$, we can find $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ ($\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$) of size at most $\binom{p+q}{p}$ in $\mathcal{O}\left(\binom{p+q}{p} t p^{\omega} + t \binom{p+q}{q}^{\omega-1}\right)$ operations over $\mathbb{F}$.*

Proof. We only show how to find in the claimed running time $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$. The proof for $\widehat{\mathcal{S}} \subseteq_{maxrep}^{q} \mathcal{S}$ is analogous, and for that case we only point out the places where the proof differs. If $t \leq \binom{k}{p}$, then we can take $\widehat{\mathcal{S}} = \mathcal{S}$. Clearly, in this case $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$. So from now onwards we always assume that $t > \binom{k}{p}$. For the proof we view the representation matrix A_M as a vector space over $\mathbb{F}$ and each set $S_i \in \mathcal{S}$ as a subspace of this vector space. For every element $e \in E$, let x_e be the corresponding k-dimensional column in A_M. Observe that each $x_e \in \mathbb{F}^k$. For each subspace $S_i \in \mathcal{S}$, $i \in \{1, \ldots, t\}$, we associate a vector $\vec{s}_i = \bigwedge_{j \in S_i} x_j$ in $\mathbb{F}^{\binom{k}{p}}$ as follows. In exterior algebra terminology, the vector $\vec{s}_i$ is a wedge product of the vectors corresponding to elements in S_i. For a set $S \in \mathcal{S}$ and $I \in \binom{[k]}{p}$, we define $s[I] = \det(A_M[I, S])$.

We also define

$$\vec{s}_i = (s_i[I])_{I \in \binom{[k]}{p}}.$$

Thus the entires of the vector $\vec{s}_i$ are the values of $\det(A_M[I, S_i])$, where I runs through all the p sized subset of rows of A_M.

Let $H_{\mathcal{S}} = (\vec{s}_1, \ldots, \vec{s}_t)$ be the $\binom{k}{p} \times t$ matrix obtained by taking $\vec{s}_i$ as columns. Now we define a weight function $w' : \mathbf{C}(H_{\mathcal{S}}) \to \mathbb{R}^+$ on the set of columns of $H_{\mathcal{S}}$. For the column s_i corresponding to $S_i \in \mathcal{S}$, we define $w'(s_i) = w(S_i)$. Let $\mathcal{W}$ be a set of columns of $H_{\mathcal{S}}$ that are linearly independent over $\mathbb{F}$, the size of $\mathcal{W}$ is equal to the $\mathsf{rank}(H_{\mathcal{S}})$ and is of minimum total weight with respect to the weight function w'. That is, $\mathcal{W}$ is a minimum weight column basis of $H_{\mathcal{S}}$. Since the row-rank of a matrix is equal to the column-rank, we have that $|\mathcal{W}| = \mathsf{rank}(H_{\mathcal{S}}) \le \binom{k}{p}$. We define $\widehat{\mathcal{S}} = \{S_\alpha \mid \vec{s}_\alpha \in \mathcal{W}\}$. Let $|\widehat{\mathcal{S}}| = \ell$. Because $|\mathcal{W}| = |\widehat{\mathcal{S}}|$, we have that $\ell \le \binom{k}{p}$. Without loss of generality, let $\widehat{\mathcal{S}} = \{S_i \mid 1 \le i \le \ell\}$ (else we can rename these sets) and $\mathcal{W} = \{\vec{s}_1 \ldots, \vec{s}_\ell\}$. The only thing that remains to show is that indeed $\widehat{\mathcal{S}} \subseteq^q_{minrep} \mathcal{S}$.

Let $S_\beta \in \mathcal{S}$ be such that $S_\beta \notin \widehat{\mathcal{S}}$. We show that if there is a set $Y \subseteq E$ of size at most q such that $S_\beta \cap Y = \emptyset$ and $S_\beta \cup Y \in \mathcal{I}$, then there exists a set $\widehat{S}_\beta \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{S}_\beta \cup Y \in \mathcal{I}$ and $w(\widehat{S}_\beta) \le w(S_\beta)$. Let us first consider the case $|Y| = q$. Since $S_\beta \cap Y = \emptyset$, it follows that $|S_\beta \cup Y| = p + q = k$. Furthermore, since $S_\beta \cup Y \in \mathcal{I}$, we have that the columns corresponding to $S_\beta \cup Y$ in A_M are linearly independent over $\mathbb{F}$; that is, $\det(A_M[\mathbf{R}(A_M), S_\beta \cup Y]) \ne 0$.

Recall that, $\vec{s}_\beta = (s_\beta[I])_{I \in \binom{[k]}{p}}$, where $s_\beta[I] = \det(A_M[I, S_\beta])$. Similarly we define $y[L] = \det(A_M[L, Y])$ and

$$\vec{y} = (y[L])_{L \in \binom{[k]}{q}} .$$

Let $\sum J = \sum_{j \in S_\beta} j$. Define

$$\gamma(\vec{s}_\beta, \vec{y}) = \sum_{I \in \binom{[k]}{p}} (-1)^{\sum I + \sum J} s_\beta[I] \cdot y[\bar{I}].$$

Since $\binom{k}{p} = \binom{k}{k-p} = \binom{k}{q}$ the above formula is well defined. Observe that by Proposition 3.1, we have that $\gamma(\vec{s}_\beta, \vec{y}) = \det(A_M[\mathbf{R}(A_M), S_\beta \cup Y]) \ne 0$. We also know that $\vec{s}_\beta$ can be written as a linear combination of vectors in $\mathcal{W} = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_\ell\}$. That is, $\vec{s}_\beta = \sum_{i=1}^\ell \lambda_i \vec{s}_i$, $\lambda_i \in \mathbb{F}$, and for some i, $\lambda_i \ne 0$. Thus,

$$
\begin{aligned}
\gamma(\vec{s}_\beta, \vec{y}) &= \sum_I (-1)^{\sum I + \sum J} s_\beta[I] \cdot y[\bar{I}] \\
&= \sum_I (-1)^{\sum I + \sum J} \left(\sum_{i=1}^\ell \lambda_i s_i[I] \right) y[\bar{I}] \\
&= \sum_{i=1}^\ell \lambda_i \left(\sum_I (-1)^{\sum I + \sum J} s_i[I] y[\bar{I}] \right) \\
&= \sum_{i=1}^\ell \lambda_i \det(A_M[\mathbf{R}(A_M), S_i \cup Y]) \qquad \text{(by Proposition 3.1)}
\end{aligned}
$$

Define

$$\mathsf{sup}(S_\beta) = \left\{ S_i \; \middle| \; S_i \in \widehat{\mathcal{S}}, \; \lambda_i \det(A_M[\mathbf{R}(A_M), S_i \cup Y])) \ne 0 \right\}.$$

Since $\gamma(\vec{s}_\beta, \vec{y}) \ne 0$, we have that $(\sum_{i=1}^\ell \lambda_i \det(A_M[\mathbf{R}(A_M), S_i \cup Y])) \ne 0$ and thus $\mathsf{sup}(S_\beta) \ne \emptyset$. Observe that for all $S \in \mathsf{sup}(S_\beta)$ we have that $\det(A_M[\mathbf{R}(A_M), S \cup Y]) \ne 0$ and thus $S \cup Y \in \mathcal{I}$. We now show that $w(S) \le w(S_\beta)$ for all $S \in \mathsf{sup}(S_\beta)$.

Claim 3.1. *For all $S \in \mathsf{sup}(S_\beta)$, $w(S) \le w(S_\beta)$.*

Proof. For contradiction assume that there exists a set $S_j \in \mathsf{sup}(S_\beta)$ such that $w(S_j) > w(S_\beta)$. Let $\vec{s_j}$ be the vector corresponding to S_j and $\mathcal{W}' = (\mathcal{W} \cup \{\vec{s_j}\}) \setminus \{\vec{s_\beta}\}$. Since $w(S_j) > w(S_\beta)$, we have that $w(\vec{s_j}) > w(\vec{s_\beta})$ and thus $w'(\mathcal{W}) > w'(\mathcal{W}')$. Now we show that $\mathcal{W}'$ is also a column basis of $H_\mathcal{S}$. This will contradict our assumption that $\mathcal{W}$ is a minimum weight column basis of $H_\mathcal{S}$. Recall that $\vec{s_\beta} = \sum_{i=1}^{\ell} \lambda_i \vec{s_i}$, $\lambda_i \in \mathbb{F}$. Since $S_j \in \mathsf{sup}(S_\beta)$, we have that $\lambda_j \neq 0$. Thus $\vec{s_j}$ can be written as linear combination of vectors in $\mathcal{W}'$. That is,

$$\vec{s_j} = \lambda_\beta \vec{s_\beta} + \sum_{i=1,i\neq j}^{\ell} \lambda'_i \vec{s_i}. \tag{1}$$

Also every vector $\vec{s_\gamma} \notin \mathcal{W}$ can be written as a linear combination of vectors in $\mathcal{W}$

$$\vec{s_\gamma} = \sum_{i=1}^{\ell} \delta_i \vec{s_i}, \quad \delta_i \in \mathbb{F}. \tag{2}$$

By substituting (1) into (2), we conclude that vector can be written as linear combination of vectors in $\mathcal{W}'$. This shows that $\mathcal{W}'$ is also a column basis of $H_\mathcal{S}$, a contradiction proving the claim. $\qquad \square$

Claim 3.1 and the discussions preceding above it show that we could take any set $S \in \mathsf{sup}(S_\beta)$ as the desired $\widehat{S}_\beta \in \widehat{\mathcal{S}}$. This shows that indeed $\widehat{\mathcal{S}} \subseteq_{minrep}^{q} \mathcal{S}$ for each Y of size q. This completes the proof for the case $|Y| = q$.

Suppose that $|Y| = q' < q$. Since M is a matroid of rank $k = p + q$, there exists a superset $Y' \in \mathcal{I}$ of Y of size q such that $S_\beta \cap Y' = \emptyset$ and $S_\beta \cup Y' \in \mathcal{I}$. This implies that there exists a set $\widehat{S} \in \widehat{\mathcal{S}}$ such that $\det(A_M[\mathbf{R}(A_M), \widehat{S} \cup Y']) \neq 0$ and $w(\widehat{S}) \leq w(S)$. Thus the columns corresponding to $\widehat{S} \cup Y$ are linearly independent.

We consider the running time of the algorithm. To make the above proof algorithmic we need to (a) compute determinants and (b) apply fast Gaussian elimination to find a minimum weight column basis. It is well known that one can compute the determinant of a $n \times n$ matrix in time $\mathcal{O}(n^\omega)$ [9]. For a rectangular matrix A of size $d \times n$ (with $d \leq n$), Bodlaender et al. [7] outline an algorithm computing a minimum weight column basis in time $\mathcal{O}(nd^{\omega-1})$. Thus given a p-family of independent sets $\mathcal{S}$ we can construct the matrix $H_\mathcal{S}$ as follows. For every set S_i, we first compute $\vec{s_i}$. To do this we compute $\det(A_M[I, S_i])$ for every $I \in \binom{[k]}{p}$. This can be done in time $\mathcal{O}(\binom{p+q}{p}p^\omega)$. Thus, we can obtain the matrix $H_\mathcal{S}$ in time $\mathcal{O}(\binom{p+q}{p}tp^\omega)$. Given matrix $H_\mathcal{S}$ we can find minimum weight basis $\mathcal{W}$ of linearly independent columns of $H_\mathcal{S}$ of total minimum weight in time $\mathcal{O}(t\binom{p+q}{p}^{\omega-1})$. Given $\mathcal{W}$, we can easily recover $\widehat{S}$. Thus, we can compute $\widehat{S} \subseteq_{minrep}^{q} \mathcal{S}$ in $\mathcal{O}\left(\binom{p+q}{p}tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$ field operations. This concludes the proof for finding $\widehat{S} \subseteq_{minrep}^{q} \mathcal{S}$. To find $\widehat{S} \subseteq_{maxrep}^{q} \mathcal{S}$, the only change we need to do in the algorithm for finding $\widehat{S} \subseteq_{minrep}^{q} \mathcal{S}$ is to find a *maximum weight column basis* $\mathcal{W}$ of $H_\mathcal{S}$. This concludes the proof. $\qquad \square$

In Theorem 3 we assumed that $\mathsf{rank}(M) = p + q$. However, one can obtain a similar result even when $\mathsf{rank}(M) > p + q$ in lieu of randomness. To do this we first need to compute the representation matrix of a k-restriction of $M = (E, \mathcal{I})$. For that we make use of Proposition 2.4. This step returns a representation of a k-restriction of $M = (E, \mathcal{I})$ with a high probability. Given this matrix, we apply Theorem 3 and arrive at the following result.

Theorem 4. *Let $M = (E, \mathcal{I})$ be a linear matroid and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets. Then there exists $\widehat{\mathcal{S}} \subseteq_{rep}^q \mathcal{S}$ of size $\binom{p+q}{p}$. Furthermore, given a representation A_M of M over a field $\mathbb{F}$, there is a randomized algorithm computing $\widehat{\mathcal{S}} \subseteq_{rep}^q \mathcal{S}$ in $\mathcal{O}\left(\binom{p+q}{p} t p^\omega + t \binom{p+q}{q}^{\omega-1}\right)$ operations over $\mathbb{F}$.*

4 Fast Computation for Representative Sets for Uniform Matroids

In this section we show that for uniform matroids one can avoid matrix multiplication computations ion order to compute representative families. The section is organized as follows. We start (Section 4.1, Theorem 5) from a relatively simple algorithm computating representative families over a uniform matroid. This algorithm is already faster than the algorithm of Theorem 1 for general matroids. In Section 4.2, Theorem 2, we give an even faster, but more complicated algorithm. Throughout this section a subfamily $\mathcal{A}' \subseteq \mathcal{A}$ of the family $\mathcal{A}$ is said to *q-represent* $\mathcal{A}$ if for every set B of size q such that there is an $A \in \mathcal{A}$ and $A \cap B = \emptyset$, there is a set $A' \in \mathcal{A}'$ such that $A' \cap B = \emptyset$.

4.1 Representative Sets using Lopsided Universal Sets

Our aim is to prove the following theorem.

Theorem 5. *There is an algorithm that given a family $\mathcal{A}$ of sets of size p over a universe U of size n and an integer q, computes in time $|\mathcal{A}| \cdot \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ a subfamily $\mathcal{A}' \subseteq A$ such that $|\mathcal{A}'| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ and $\mathcal{A}'$ q-represents $\mathcal{A}$.*

The main tool in our proof of Theorem 5 is a generalization of the notion of *n,k-universal* families. A family $\mathcal{F}$ of sets over a universe U is an *n-k-universal* family if for every set $A \subseteq \binom{U}{k}$ and every subset $A' \subseteq A$ there is some set $F \in \mathcal{F}$ whose intersection $F \cap A$ with A is exactly equal to A'. Naor et al. [38] show that given n and k one can construct an *n-k*-universal family $\mathcal{F}$ of size $2^{k+o(k)} \cdot \log n$ in time $2^{k+o(k)} \cdot n \log n$.

We tweak the notion of universal families as follows. We will say that a family $\mathcal{F}$ of sets over a universe U is an *n-p-q-lopsided-universal* family if for every $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$ there is an $F \in \mathcal{F}$ such that $A \subseteq F$ and $B \cap F = \emptyset$. An alternative definition that is easily seen to be equivalent is that $\mathcal{F}$ is *n,p-q*-lopsided-universal if for every subset $A \subseteq \binom{U}{p+q}$ and every subset $A' \in \binom{A}{p}$, there is an $F \in \mathcal{F}$ such that $F \cap A = A'$. From the second definition it follows that a *n-$(p+q)$*-universal family is also *n-p-q*-lopsided-universal. Thus the construction of Naor et al. [38] of universal set families also gives a construction of *n-p-q*-lopsided universal family of size $2^{p+q+o(p+q)} \cdot \log n$, running in time $2^{p+q+o(p+q)} \cdot n \log n$. It turns out that by slightly changing the construction of Naor et al. [38], one can prove the following result.

Lemma 4.1. *There is an algorithm that given n, p and q constructs a n-p-q-lopsided-universal family $\mathcal{F}$ of size $\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ in time $\mathcal{O}(\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot n \log n)$.*

We do not give a stand-alone proof of Lemma 4.1, however Lemma 4.1 is a direct corollary of Lemma 4.2 proved in Section 4.2. We will now show how to use the lemma to prove Theorem 5.

Proof of Theorem 5. The algorithm starts by constructing a n-p-q-lopsided universal family $\mathcal{F}$ as guaranteed by Lemma 4.1. If $|\mathcal{A}| \leq |\mathcal{F}|$ the algorithm outputs $\mathcal{A}$ and halts. Otherwise it builds the set $\mathcal{A}'$ as follows. We assume that $n \leq |\mathcal{A}| \cdot p$, since otherwise some elements of the universe are not contained in any set in $\mathcal{A}$ and may safely be ignored. Initially $\mathcal{A}'$ is equal to $\emptyset$

and all sets in $\mathcal{F}$ are marked as unused. The algorithm goes through every $A \in \mathcal{A}$ and unused sets $F \in \mathcal{F}$. If an unused set $F \in \mathcal{F}$ is found such that $A \subseteq F$, the algorithm marks F as used, inserts A into $\mathcal{A}'$ and proceeds to the next set in $\mathcal{A}$. If no such set F is found the algorithm proceeds to the next set in $\mathcal{A}$ without inserting A into $\mathcal{A}'$.

The size of $\mathcal{A}'$ is upper bounded by $|\mathcal{F}| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ since every time a set is added to $\mathcal{A}'$ an unused set in $\mathcal{F}$ is marked as used. For the running time analysis, constructing $\mathcal{F}$ takes time $\binom{p+q}{p} \cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot n \log n$. Then we run through all of $\mathcal{F}$ for each set $A \in \mathcal{A}$, spending time $|\mathcal{A}| \cdot |\mathcal{F}| \cdot (p+q)^{\mathcal{O}(1)}$, which is at most $|\mathcal{A}| \cdot \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$. Thus in total the running time is bounded by $|\mathcal{A}| \cdot \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$.

Finally we need to argue that $\mathcal{A}'$ q-represents $\mathcal{A}$. Consider any set $A \in \mathcal{A}$ and B such that $|B| = q$ and $A \cap B = \emptyset$. If $A \in \mathcal{A}'$ we are done, so assume that $A \notin \mathcal{A}'$. Since $\mathcal{F}$ is n-p-q-lopsided universal there is a set $F \in \mathcal{F}$ such that $A \subseteq F$ and $F \cap B = \emptyset$. Since $A \notin \mathcal{A}'$ we know that F was already marked as used when A was considered by the algorithm. When the algorithm marked F as used it also inserted a set A' into $\mathcal{A}'$. For the insertion to be made, F must satisfy $A' \subseteq F$. But then $A' \cap B = \emptyset$, completing the proof. □

One of the factors that drive up the running time of the algorithm in Theorem 5 is that one needs to consider all of $\mathcal{F}$ for each set $A \in \mathcal{A}$. Doing some computations it is possible to convince oneself that in a n-p-q-lopsided universal family $\mathcal{F}$ the number of sets $F \in \mathcal{F}$ that contain a fixed set A of size p should be approximately $|\mathcal{F}| \cdot \left(\frac{p}{p+q}\right)^p$. Thus, if we could only make sure that this estimation is in fact correct for every $A \in \mathcal{A}$, *and* we could make sure that for a given $A \in \mathcal{A}$ we can list all of the sets in $\mathcal{F}$ that contain A without having to go through the sets that don't, then we could speed up our algorithm by a factor $\left(\frac{p+q}{p}\right)^p$. This is exactly the strategy behind the main theorem of Section 4.2.

4.2 Representative Sets using Separating Collections

The goal of this section is to prove the following theorem.

Theorem 2 (restated). *There is an algorithm that given a family $\mathcal{A}$ of sets of size p over a universe U of size n and an integer q, computes in time $\mathcal{O}(|\mathcal{A}| \cdot (\frac{p+q}{q})^q \cdot \log n)$ a subfamily $\mathcal{A}' \subseteq \mathcal{A}$ such that $|\mathcal{A}'| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ and $\mathcal{A}'$ q-represents $\mathcal{A}$.*

We say that a family $\mathcal{F}$ *separates* a set A from a set B if there is an $F \in \mathcal{F}$ such that $A \subseteq F$ and $B \cap F = \emptyset$. The basic idea behind the proof of Theorem 2 is a construction of a small family $\mathcal{F}$ that separates every $A \in \mathcal{A}$ from every set B of size q such that $A \cap B = \emptyset$. We now define *separating collections*, which is basically a term that allows us to speak about how small the computed family $\mathcal{F}$ is, and how efficiently we can compute it.

A n-p-q-*separating collection* $\mathcal{C}$ is a pair $(\mathcal{F}, \chi)$, where $\mathcal{F}$ is a family of sets over a universe U of size n and χ is a function from $\binom{U}{p}$ to $2^{\mathcal{F}}$ such that the following two properties are satisfied; (a) for every $A \in \binom{U}{p}$ and $F \in \chi(A)$, $A \subseteq F$, (b) for every $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$, $\chi(A)$ separates A from B. The *size* of $(\mathcal{F}, \chi)$ is $|\mathcal{F}|$, whereas the *max degree* of $(\mathcal{F}, \chi)$ is $\max_{A \in \binom{U}{p}} |\chi(A)|$.

A *construction* of separating collections is a data structure, that given n, p and q initializes and outputs a family $\mathcal{F}$ of sets over the universe U of size n. After the initialization one can query the data structure by giving it a set $A \in \binom{U}{p}$, the data structure then outputs a family $\chi(A) \subseteq 2^{\mathcal{F}}$. Together the pair $\mathcal{C} = (\mathcal{F}, \chi)$ computed by the data structure should form a n-p-q-separating collection.

We call the time the data structure takes to initialize and output $\mathcal{F}$ the *initialization time*. The *query time* of the data structure is the maximum time the data structure uses to compute $\chi(A)$ over all $A \in \binom{U}{p}$. The initialization time and query time of the data structure and the size

and degree of $\mathcal{C}$ are functions of n, p and q. The initialization time is denoted by $\tau_I(n,p,q)$, the query time by $\tau_Q(n,p,q)$, the resulting size of $\mathcal{C}$ is denoted by $\zeta(n,p,q)$, while the degree of $\mathcal{C}$ is denoted by $\Delta(n,p,q)$. The main technical component in the proof of Theorem 2 is the following lemma.

Lemma 4.2. *There is a construction of separating collections with the following parameters*

- *size* $\zeta(n,p,q) \leq \binom{p+q}{p} \cdot 2^{\mathcal{O}\left(\frac{p+q}{\log\log(p+q)}\right)} \cdot \log n$,

- *initialization time* $\tau_I(n,p,q) \leq \binom{p+q}{p} \cdot 2^{\mathcal{O}\left(\frac{p+q}{\log\log(p+q)}\right)} \cdot n \log n$,

- *degree* $\Delta(n,p,q) \leq \left(\frac{p+q}{q}\right)^q \cdot 2^{\mathcal{O}\left(\frac{p+q}{\log\log(p+q)}\right)} \cdot \log n$, *and*

- *query time* $\tau_Q(n,p,q) \leq \left(\frac{p+q}{q}\right)^q \cdot 2^{\mathcal{O}\left(\frac{p+q}{\log\log(p+q)}\right)} \cdot \log n$.

We will first prove how Lemma 4.2 yields a proof of Theorem 2. The rest of the section contains a proof of Lemma 4.2.

Proof of Theorem 2. The algorithm starts by constructing a n-p-q-separating collection $(\mathcal{F}, \chi)$ as guaranteed by Lemma 4.2. If $|\mathcal{A}| \leq |\mathcal{F}|$ the algorithm outputs $\mathcal{A}$ and halts. Otherwise it builds the set $\mathcal{A}'$ as follows. Initially $\mathcal{A}'$ is equal to $\emptyset$ and all sets in $\mathcal{F}$ are marked as unused. The algorithm goes through every $A \in \mathcal{A}$ and queries the separating collection to get the set $\chi(A)$. It then looks for a set $F \in \chi(A)$ that is not yet marked as used. The first time such a set F is found the algorithm marks F as used, inserts A into $\mathcal{A}'$ and proceeds to the next set in $\mathcal{A}$. If no such set F is found the algorithm proceeds to the next set in $\mathcal{A}$ without inserting A into $\mathcal{A}'$.

The size of $\mathcal{A}'$ is upper bounded by $|\mathcal{F}| \leq \binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$ since every time a set is added to $\mathcal{A}'$ an unused set in $\mathcal{F}$ is marked as used. For the running time analysis, the initialization of $(\mathcal{F}, \chi)$ takes time $\binom{p+q}{p} \cdot 2^{\mathcal{O}\left(\frac{p+q}{\log\log(p+q)}\right)} \cdot n \log n$. For each element $A \in \mathcal{A}$ the algorithm first queries $\chi(A)$, using time $\left(\frac{p+q}{q}\right)^q \cdot 2^{\mathcal{O}\left(\frac{p+q}{\log\log(p+q)}\right)} \cdot \log n$. Then it goes through all sets in $\chi(A)$ and checks whether they have already been marked as used, taking time $\left(\frac{p+q}{q}\right)^q \cdot 2^{\mathcal{O}\left(\frac{p+q}{\log\log(p+q)}\right)} \cdot \log n$. Thus in total, the running time is bounded by $\mathcal{O}(|\mathcal{A}| \cdot \left(\frac{p+q}{q}\right)^q \cdot 2^{o(p+q)} \cdot \log n)$ as claimed.

Finally we need to argue that $\mathcal{A}'$ q-represents $\mathcal{A}$. Consider any set $A \in \mathcal{A}$ and B such that $|B| = q$ and $A \cap B = \emptyset$. If $A \in \mathcal{A}'$ we are done, so assume that $A \notin \mathcal{A}'$. Since $\chi(A)$ separates A from B there is a set $F \in \chi(A)$ such that $A \subseteq F$ and $F \cap B = \emptyset$. Since $A \notin \mathcal{A}'$ we know that F was marked as used when A was considered by the algorithm. When the algorithm marked F as used it also inserted a set A' into $\mathcal{A}'$, with the property that $F \in \chi(A')$. Thus $A' \subseteq F$ and hence $A' \cap B = \emptyset$. But $A' \in \mathcal{A}'$, completing the proof. $\qquad\square$

We now turn to the proof of Lemma 4.2. The proof is based on the *splitters* technique of Naor et al. [38]. We start by giving a construction of separating collections with good bounds on the size and the degree, quite reasonable query time but with really slow initialization time.

Lemma 4.3. *There is a construction of separating collections with*

- *size* $\zeta(n,p,q) = \mathcal{O}(\binom{p+q}{p} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$,
- *initialization time* $\tau_I(n,p,q) = \mathcal{O}(\binom{2^n}{\zeta(n,p,q)} \cdot n^{\mathcal{O}(p+q)})$,
- *degree* $\Delta(n,p,q) = \mathcal{O}(\left(\frac{p+q}{q}\right)^q \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$, *and*
- *query time* $\tau_Q(n,p,q) = \mathcal{O}(\binom{p+q}{p} \cdot n^{\mathcal{O}(1)})$.

Proof. We start by giving a randomized algorithm that with positive probability constructs a n-p-q-separating collection $\mathcal{C} = (\mathcal{F}, \chi)$ with the desired size and degree parameters. We will then discuss how to deterministically compute such a $\mathcal{C}$ within the required time bound. Set $t = \frac{(p+q)^{p+q}}{p^p q^q} \cdot (p+q+1) \log n$ and construct the family $\mathcal{F} = \{F_1, \ldots, F_t\}$ as follows. Each set F_i is random subset of U, where each element of U is inserted into F_i with probability $\frac{p}{p+q}$. Distinct elements are inserted (or not) into F_i independently, and the construction of the different sets in $\mathcal{F}$ is also independent. For each $A \in \binom{U}{p}$ we set $\chi(A) = \{F \in \mathcal{F} \ : \ A \subseteq F\}$.

The size of $\mathcal{F}$ is within the required bounds by construction, as $\frac{(p+q)^{p+q}}{p^p q^q} \cdot (p+q+1) \log n \leq \binom{p+q}{p} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n$. We now argue that with positive probability $(\mathcal{F}, \chi)$ is indeed a n-p-q-separating collection, and that the degree of $\mathcal{C}$ is within the required bounds as well. For a fixed set $A \in \binom{U}{p}$, set $B \in \binom{U \setminus A}{q}$, and integer $i \leq t$, we consider the probability that $A \subseteq F_i$ and $B \cap F_i = \emptyset$. This probability is

$$\left(\frac{p}{p+q}\right)^p \cdot \left(\frac{q}{p+q}\right)^q = \frac{(p+q)^{p+q}}{p^p q^q}.$$

Since each F_i is constructed independently from the other sets in $\mathcal{F}$, the probability that *no F_i* satisfies $A \subseteq F_i$ and $B \cap F_i = \emptyset$ is

$$\left(1 - \frac{p^p q^q}{(p+q)^{p+q}}\right)^t \leq e^{-(p+q+1)\log n} = \frac{1}{n^{p+q+1}}.$$

There are $\binom{n}{p}$ choices for $A \in \binom{U}{p}$ and $\binom{n-p}{q}$ choices for $B \in \binom{U \setminus A}{q}$, therefore the union bound yields that the probability that there exists an $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$ such that $\mathcal{F}$ does not separate A from B is at most $\frac{1}{n^{p+q+1}} \cdot n^{p+q} = \frac{1}{n}$. Since $\chi(A)$ contains all the sets in $\mathcal{F}$ that contain A, $\chi(A)$ separates A from B whenever $\mathcal{F}$ does.

We also need to upper bound the max degree of $\mathcal{C}$. For every $A \in \binom{U}{p}$, $|\chi(A)|$ is a random variable. For a fixed $A \in \binom{U}{p}$ and $i \leq t$ the probability that $A \subseteq F_i$ is exactly $(\frac{p}{p+q})^p$. Hence $|\chi(A)|$ is the sum of t independent $0/1$ variables that each take value 1 with probability $(\frac{p}{p+q})^p$. Hence the expected value of $|\chi(A)|$ is

$$E[|\chi(A)|] = t \cdot \left(\frac{p}{p+q}\right)^p = \left(\frac{p+q}{q}\right)^q \cdot (p+q+1) \log n.$$

Standard Chernoff bounds [34, Theorem 4.4] show that the probability that $|\chi(A)|$ is at least $6E[|\chi(A)|]$ is upper bounded by $2^{-6E[|\chi(A)|]} \leq \frac{1}{n^{p+q+1}}$. There are at most $\binom{n}{p}$ choices for $A \in \binom{U}{p}$. Hence the union bound yields that the probability that there exists an $A \in \binom{U}{p}$ such that $|\chi(A)| > 6E[|\chi(A)|]$ is upper bounded by $\frac{1}{n}$. Thus $\mathcal{C}$ is a family of n-p-q-balanced universal sets with the desired size and degree parameters with probability at least $1 - \frac{2}{n} > 0$. The degenerate case that $1 - \frac{2}{n} \leq 0$ is handled by the family $\mathcal{F}$ containing all (at most four) subsets of U.

To construct $\mathcal{F}$ within the stated initialization time bound, it is sufficient to try all families $\mathcal{F}$ of size t and for each of the $\binom{2^n}{\zeta(n,p,q)}$ guesses test whether it is indeed a family of n-p-q-balanced universal sets in time $\mathcal{O}(t \cdot n^{\mathcal{O}(p+q)}) = \mathcal{O}(n^{\mathcal{O}(p+q)})$.

For the queries, we need to give an algorithm that computes given A computes $\chi(A)$, under the assumption that $\mathcal{F}$ has already has been computed in the initialization step. This is easily done within the stated running time bound by going through every set $F \in \mathcal{F}$, checking whether $A \subseteq F$, and if so, inserting F into $\chi(A)$. This concludes the proof. $\square$

We will now work towards improving the time bounds of Lemma 4.3. To that end we will need a construction of *k-perfect hash functions* by Alon et al. [1]. A family of functions $f_1, \ldots, f_t$

from a universe U of size n to a universe of size r is a k-perfect family of hash functions if for every set $S \subseteq U$ such that $|S| = k$ there exists an i such that the restriction of f_i to S is injective. Alon et al. [1] give very efficient constructions of k-perfect families of hash functions from a universe of size n to a universe of size k^2

Proposition 4.1 ([1])**.** *For any universe U of size n there is a k-perfect family $f_1, \dots, f_t$ of hash functions from U to $\{1, 2, \dots, k^2\}$ with $t = \mathcal{O}(k^{\mathcal{O}(1)} \cdot \log n)$. Such a family of hash functions can be constructed in time $\mathcal{O}(k^{\mathcal{O}(1)} n \log n)$.*

We now use Proposition 4.1 to give a *universe reduction* lemma that allows us to reduce the construction of separating collections to constructions of separating collections over a small universe.

Lemma 4.4. *If there is a construction of n-p-q-separating collections with initialization time $\tau_I(n, p, q)$, query time $\tau_Q(n, p, q)$, producing a n-p-q-separating collection with size $\zeta(n, p, q)$ and degree $\Delta(n, p, q)$, then there is a construction of n-p-q-separating collections using*

- *size $\zeta'(n, p, q) \leq \zeta((p + q)^2, p, q) \cdot (p + q)^{\mathcal{O}(1)} \cdot \log n$,*

- *initialization time $\tau_I'(n, p, q) \leq \mathcal{O}(\tau_I((p + q)^2, p, q) + \zeta((p + q)^2, p, q) \cdot (p + q)^{\mathcal{O}(1)} \cdot n \log n)$,*

- *degree $\Delta'(n, p, q) \leq \Delta((p + q)^2, p, q) \cdot (p + q)^{\mathcal{O}(1)} \cdot \log n$, and*

- *query time $\tau_Q'(n, p, q) \leq \mathcal{O}((\tau_Q((p + q)^2, p, q) + \Delta((p + q)^2, p, q)) \cdot (p + q)^{\mathcal{O}(1)} \cdot \log n)$.*

Proof. We give a construction of n-p-q-separating collections with initialization time, query time, size and degree τ_I', τ_Q', ζ' and Δ' respectively using the construction with initialization time, query time, size and degree τ_I, τ_Q, ζ and Δ as a black box.

We first describe the initialization of the data structure. Given n, p and q, we construct using Proposition 4.1 a $(p + q)$-perfect family $f_1, \dots f_t$ of hash functions from the universe U to $\{1, 2, \dots, k^2\}$. The construction takes time $\mathcal{O}((p + q)^{\mathcal{O}(1)} n \log n)$ and $t \leq (p + q)^{\mathcal{O}(1)} \cdot \log n$. We will store these hash functions in memory.

For a set $S \subseteq U$, by $f_i(S)$ we will mean $\{f_i(s) \ : \ s \in S\}$. Similarly for every $S \subseteq \{1, \dots, (p + q)^2\}$, by $f_i^{-1}(S)$ we will mean $\{s \in U \ : \ f(s) \in S\}$. For a family $\mathcal{Z}$ of sets over U, by $f_i(\mathcal{Z})$ we will mean $\{f_i(S) \ : \ S \in \mathcal{Z}\}$. Finally, for a family $\mathcal{Z}$ of sets over $\{1, \dots, (p + q)^2\}$, by $f_i^{-1}(\mathcal{Z})$ we will mean $\{f_i^{-1}(S) \ : \ S \in \mathcal{Z}\}$.

We first use the given black box construction for $(p + q)^2$-p-q-separating collections over the universe $\{1, \dots, (p + q)^2\}$. This construction computes the separating collection $(\hat{\mathcal{F}}, \hat{\chi})$. We run the initialization algorithm of this construction and store the family $\hat{\mathcal{F}}$ in memory. We then set

$$\mathcal{F} = \bigcup_{i \leq t} f_i^{-1}(\hat{\mathcal{F}}).$$

We spent $\mathcal{O}((p + q)^{\mathcal{O}(1)} n \log n)$ time to construct the a $(p + q)$-perfect family of hash functions, $\mathcal{O}(\tau((p + q)^2, p, q))$ to construct $\hat{\mathcal{F}}$ of size $\zeta((p + q)^2, p, q)$, and $\mathcal{O}(\zeta((p + q)^2, p, q) \cdot (p + q)^{\mathcal{O}(1)} \cdot n \log n)$ time to construct $\mathcal{F}$ from $\hat{\mathcal{F}}$ and the family of perfect hash functions. Thus the upper bound on $\tau_I'(n, p, q)$ follows. Furthermore, $|\mathcal{F}| \leq |\hat{\mathcal{F}}| \cdot (p + q)^{\mathcal{O}(1)} \cdot \log n$, yielding the claimed bound for ζ'.

We now define $\chi(A)$ for every $A \in \binom{U}{p}$ and describe the query algorithm. For every $A \in \binom{U}{p}$ we let

$$\chi(A) = \bigcup_{\substack{i \leq t \\ |f_i(A)| = |A|}} f_i^{-1}(\hat{\chi}(f_i(A))).$$

18

Since $f_i(A) \subseteq \hat{F}$ for every $\hat{F} \in \hat{\chi}(f_i(A))$ it follows that $A \in F$ for every $F \in \chi(A)$. Furthermore we can bound $|\chi(A)|$ as follows

$$|\chi(A)| \leq \sum_{\substack{i \leq t \\ |f_i(A)|=|A|}} |\hat{\chi}(f_i(A))| \leq \Delta((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n.$$

Thus the claimed bound for Δ' follows. To compute $\chi(A)$ we go over every $i \leq t$ and check whether f_i is injective on A. This takes time $\mathcal{O}((p+q)^{\mathcal{O}(1)} \cdot \log n)$. For each i such that f_i is injective on A, we compute $f_i(A)$ and then $\hat{\chi}(f_i(A))$ in time $\mathcal{O}(\tau_Q((p+q)^2, p, q) + p + q)$. Then we compute $f_i^{-1}(\hat{\chi}(f_i(A)))$ in time $\mathcal{O}(|\hat{\chi}(f_i(A))| \cdot (p+q)^{\mathcal{O}(1)}) = \mathcal{O}(\Delta((p+q)^2, p, q) \cdot (p+q)^{\mathcal{O}(1)})$ and add this set to $\chi(A)$. As we need to do this $\mathcal{O}((p+q)^{\mathcal{O}(1)} \cdot \log n)$ times, the total time to compute $\chi(A)$ is upper bounded by $\mathcal{O}((\tau_Q((p+q)^2, p, q) + \Delta((p+q)^2, p, q)) \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$, yielding the claimed upper bound on τ_Q'.

It remains to argue that $(\mathcal{F}, \chi)$ is in fact a n-p-q-separating collection. Consider a set $A \in \binom{U}{p}$ and $\binom{B \in (U \setminus A)}{q}$. We need to show that $\chi(A)$ separates A from B. Since $f_1, \ldots, f_t$ is a $(p+q)$-perfect family of hash functions, there is an i such that f_i is injective on $A \cup B$. Since $(\hat{\mathcal{F}}, \hat{\chi})$ is a $(p+q)^2$-p-q-separating collection, we have that $\hat{\chi}(f_i(A))$ separates $f_i(A)$ from $f_i(B)$. Thus $f_i^{-1}(\hat{\chi}(f_i(A)))$ separates A from B. Since f_i is injective on A it follows that $f_i^{-1}(\hat{\chi}(f_i(A))) \subseteq \chi(A)$, and hence $\chi(A)$ separates A from B, concluding the proof. $\qquad\square$

We now give a *splitting lemma*, which allows us to reduce the problem of finding n-p-q-separating collections to the same problem, but with much smaller values for p and q. To that end we need some definitions.

A *partition* of U is a family $\mathcal{U}_P = \{U_1, U_2, \ldots U_t\}$ of sets over U such that $U_i \cap U_j = \emptyset$ for every $i \neq j$ and $U = \bigcup_{i \leq t} U_i$. Each of the sets U_i are called the *parts* of the partition. A *consecutive partition* of $\{1, \ldots, n\}$ is a partition $\mathcal{U}_P = \{U_1, U_2, \ldots U_t\}$ of $\{1, \ldots, n\}$ such that for every integer $i \leq t$ and integers $1 \leq x \leq y \leq z$, if $x \in U_i$ and $z \in U_i$ then $y \in U_i$ as well. In other words, in a consecutive partition each part is a consecutive interval of integers. For every integer t, let $\mathscr{P}_t$ denote the collection of all consecutive partitions of $\{1, \ldots, n\}$ with exaclty t parts. We do not demand that all of the parts in a partition in $\mathscr{P}_t$ are non-empty. Simple counting arguments show that for every t, $|\mathscr{P}_t| = \binom{n+t-1}{t-1}$.

We will denote by $\mathcal{Z}_{s,t}^p$ the set of all t-tuples $(p_1, p_2, \ldots, p_t)$ of integers such that $\sum_{i \leq t} p_i = p$ and $0 \leq p_i \leq s$ for all i. Clearly $|\mathcal{Z}_{s,t}^p| \leq \binom{p+t-1}{t-1}$, since this counts all the ways of writing p as a sum of t non-negative integers, without considering the upper bound on each one.

Lemma 4.5. *For any p, q let $s = \lfloor (\log(p+q))^2 \rfloor$ and $t = \lceil \frac{p+q}{s} \rceil$. If there is a construction of n-p-q-separating collections with initialization time $\tau_I(n, p, q)$, query time $\tau_Q(n, p, q)$, producing a n-p-q-separating collection with size $\zeta(n, p, q)$ and degree $\Delta(n, p, q)$, then there is a construction of n-p-q-separating collections using*

- *size $\zeta'(n, p, q) \leq |\mathscr{P}_t| \cdot \sum_{(p_1, \ldots, p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} \zeta(n, p_i, s - p_i)$,*

- *initialization time $\tau_I'(s, p, q) \leq O\left(\left(\sum_{\hat{p} \leq s} \tau_I(n, \hat{p}, s - \hat{p})\right) + \zeta'(n, p, q) \cdot n^{\mathcal{O}(1)}\right)$,*

- *degree $\Delta'(n, p, q) \leq |\mathscr{P}_t| \cdot \max_{(p_1, \ldots, p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} \Delta(n, p_i, s - p_i)$, and*

- *query time $\tau_Q'(n, p, q) \leq O\left(\Delta'(n, p, q) \cdot n^{\mathcal{O}(1)} + |\mathscr{P}_t| \cdot t \cdot \sum_{\hat{p} \leq s} \tau_Q(n, \hat{p}, s - \hat{p})\right)$.*

Proof. Set $s = \lfloor (\log(p+q))^2 \rfloor$, $t = \lceil \frac{p+q}{s} \rceil$ and $\tilde{q} = st - p$. We will give a construction of n-p-$\tilde{q}$-separating collections with initialization time, query time, size and degree within the claimed

bounds for $\tau_I'(n,p,q)$, $\tau_Q'(n,p,q)$, $\zeta'(n,p,q)$ and $\Delta'(n,p,q)$ respectively. In the construction we will be using the construction with initialization time, query time, size and degree τ_I, τ_Q, ζ and Δ as a black box. Since $\tilde{q} \geq q$, a n-p-$\tilde{q}$-separating collection is also a n-p-q-separating collection. We may assume without loss of generality that $U = \{1, \ldots, n\}$.

Our algorithm runs for every $\hat{p}$ between 0 and s the initialization of the given construction of n-$\hat{p}$-$(s-\hat{p})$-separating collections. We will refer by $(\hat{\mathcal{F}}_{\hat{p}}, \hat{\chi}_{\hat{p}})$ to the separating collection constructed for $\hat{p}$. For each $\hat{p}$ the initialization of the construction outputs the family $\hat{\mathcal{F}}_{\hat{p}}$.

We need to define a few operations on families of sets. For a family $\mathcal{A}$ over U and subset $U' \subseteq U$ we define

$$\mathcal{A} \sqcap U' = \{A \cap U \ : \ A \in \mathcal{A}\}.$$

For two families $\mathcal{A}$ and $\mathcal{B}$ over (subsets of) U, we define

$$\mathcal{A} \circ \mathcal{B} = \{A \cup B \ : \ A \in \mathcal{A} \land B \in \mathcal{B}\}.$$

We now define $\mathcal{F}$ as follows.

$$\mathcal{F} = \bigcup_{\substack{\{U_1,\ldots,U_t\}\in\mathscr{P}_t \\ (p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p}} (\hat{\mathcal{F}}_{p_1} \sqcap U_1) \circ (\hat{\mathcal{F}}_{p_2} \sqcap U_2) \circ \ldots \circ (\hat{\mathcal{F}}_{p_t} \sqcap U_t) \tag{3}$$

It follows directly from the definition of $\mathcal{F}$ that $|\mathcal{F}|$ is within the claimed bound for $\zeta'(n,p,q)$. For the initialization time, the algorithm spends $\mathcal{O}(\sum_{\hat{p}\leq s} \tau_I(n,\hat{p},s-\hat{p}))$ time to initialize the constructions of the n-$\hat{p}$-$(s-\hat{p})$-separating collections. At that point the algorithm can output the entries of $\mathcal{F}$ one set at a time by using (3), spending $\mathcal{O}(n^{\mathcal{O}(1)})$ time per output set.

For every set $A \in \binom{U}{p}$ we define $\chi(A)$ as follows.

$$\chi(A) = \bigcup_{\substack{\{U_1,\ldots,U_t\}\in\mathscr{P}_t \ s.t \\ \forall U_i \ : \ p_i=|U_i\cap A|\leq s}} \Big[(\hat{\chi}_{p_1}(A \cap U_1) \sqcap U_1) \circ (\hat{\chi}_{p_2}(A \cap U_2) \sqcap U_2) \circ \ldots \tag{4}$$

$$\ldots \circ (\hat{\chi}_{p_t}(A \cap U_t) \sqcap U_t) \Big]$$

It follows directly from the definition of $\chi(A)$ that $|\chi(A)|$ is within the claimed bound for $\Delta'(n,p,q)$. We now describe how queries $\chi(A)$ can be answered, and analyze how much time it takes. Given A we will compute $\chi(A)$ using (4). For each $\{U_1,\ldots,U_t\} \in \mathscr{P}_t$ such that $p_i = |U_i \cap A| \leq s$ for all $i \leq t$, we proceed as follows. First we compute $\hat{\chi}_{p_i}(A \cap U_i)$ for each $i \leq t$, spending in total $\mathcal{O}(\sum_{i\leq t} \tau_Q(n,p_i,s-p_i))$ time. Now we add $(\hat{\chi}_{p_1}(A\cap U_1)\sqcap U_1)\circ(\hat{\chi}_{p_2}(A\cap U_2)\sqcap U_2)\circ\ldots\circ(\hat{\chi}_{p_t}(A\cap U_t)\sqcap U_t)$ to $\chi(A)$, spending $n^{\mathcal{O}(1)}$ time per set that is added to $\chi(A)$, yielding the bound below for $\tau_Q'(n,p,q)$.

$$\tau_Q'(n,p,q) \leq \mathcal{O}\Big(\Delta'(n,p,q) \cdot n^{\mathcal{O}(1)} + \sum_{\substack{\{U_1,\ldots,U_t\}\in\mathscr{P}_t \ s.t \\ \forall U_i \ : \ p_i=|U_i\cap A|\leq s}} \big[\sum_{i\leq t} \tau_Q(n,p_i,s-p_i) \big] \Big)$$

$$\leq \mathcal{O}\Big(\Delta'(n,p,q) \cdot n^{\mathcal{O}(1)} + |\mathscr{P}_t| \cdot \max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \big(\sum_{i\leq t} \tau_Q(n,p_i,s-p_i) \big) \Big)$$

$$\leq \mathcal{O}\Big(\Delta'(n,p,q) \cdot n^{\mathcal{O}(1)} + |\mathscr{P}_t| \cdot t \cdot \sum_{\hat{p}\leq s} \tau_Q(n,\hat{p},s-\hat{p}) \Big)$$

We now need to argue that $(\mathcal{F}, \chi)$ are in fact a n-p-$\tilde{q}$-separating collection. Consider two sets $A \in \binom{U}{p}$ and $B \in \binom{U\setminus A}{\tilde{q}}$. There exists a consecutive partition $\{U_1,\ldots,U_t\} \in \mathscr{P}_t$ of U such that for every $i \leq t$ we have that $|(A \cup B) \cap U_i| = \frac{p+\tilde{q}}{t} = s$. For each $i \leq t$ set $p_i = |A \cap U_i|$ and

20

$q_i = |B \cap U_i| = s - p_i$. For every $i \leq t$ the pair $(\hat{\mathcal{F}}_{p_i}, \hat{\chi}_{p_i})$ form a n-p_i-q_i-separating collection. Hence $\hat{\chi}_{p_i}(A \cap U_i)$ separates $A \cap U_i$ from $B \cap U_i$. Let $F_i \in \hat{\chi}_{p_i}(A \cap U_i)$ be such that $A \cap U_i \subseteq F_i$ and $F_i \cap B \cap U_i = \emptyset$. Let $F = \bigcup_{i \leq t}(F_i \cap U_i)$. Then $A \subseteq F$ and $B \cap F = \emptyset$. The construction of $\chi(A)$ ensures that $F \in \chi(A)$ and thus $\chi(A)$ separates A from B, completing the proof. $\qquad \square$

We are now ready to prove Lemma 4.2.

Proof of Lemma 4.2. The structure of the proof is simple; first we create a construction using Lemma 4.3. Applying Lemma 4.4 gives a new, second, construction. We then make more constructions by repeatedly applying Lemmata 4.5 and 4.4. Specifically the third construction is obtained by obtaining Lemma 4.5 to the second, the fourth by applying Lemma 4.4 to the third, the fifth by applying Lemma 4.5 to the fourth and the sixth and final construction is obtained by applying Lemma 4.4 to the fifth. The bulk of the work is to verify that the respective constructions indeed have the claimed parameters. We now proceed with the formal proof.

Apply Lemma 4.3 and get a construction of n-p-q-separating collections with the following parameters.

- size $\zeta^1(n, p, q) \leq \mathcal{O}(\binom{p+q}{p} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$,

- initialization time $\tau_I^1(n, p, q) \leq \mathcal{O}(\binom{2^n}{\zeta(n,p,q)} \cdot n^{\mathcal{O}(p+q)}) \leq 2^{n(p+q)^p}$,

- degree $\Delta^1(n, p, q) \leq \mathcal{O}((\frac{p+q}{q})^q \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$, and

- query time $\tau_Q^1(n, p, q) \leq \mathcal{O}(\binom{p+q}{p} \cdot n^{\mathcal{O}(1)}) \leq n^{\mathcal{O}(p+q)}$.

We apply Lemma 4.4 to this construction and get a new construction with the following parameters

- size $\zeta^2(n, p, q) \leq \mathcal{O}(\binom{p+q}{p} \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$,

- initialization time $\tau_I^2(n, p, q) \leq \mathcal{O}(2^{(p+q)^{(p+2)}} + \binom{p+q}{p} \cdot (p+q)^{\mathcal{O}(1)} \cdot n \log n)$,

- degree $\Delta^2(n, p, q) \leq \mathcal{O}((\frac{p+q}{q})^q \cdot (p+q)^{\mathcal{O}(1)} \cdot \log n)$, and

- query time $\tau_Q^2(n, p, q) \leq \mathcal{O}((p+q)^{\mathcal{O}(p+q)} \cdot \log n)$.

We now apply Lemma 4.5 to this construction. Recall that in Lemma 4.5 we set $s = \lfloor (\log(p+q))^2 \rfloor$ and $t = \lceil \frac{p+q}{s} \rceil$. This yields a new construction of separating collections, with size

$$
\begin{aligned}
\zeta^3(n, p, q) &\leq |\mathscr{P}_t| \cdot \sum_{(p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} \zeta^2(n, p_i, s - p_i) \\
&\leq 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})} \cdot \max_{(p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} \binom{s}{p_i} \cdot (s)^{\mathcal{O}(1)} \cdot \log n \\
&\leq 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})} \cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})} \cdot \max_{(p_1,\ldots,p_t) \in \mathcal{Z}_{s,t}^p} \prod_{i \leq t} \binom{s}{p_i} \\
&\leq \binom{p+q}{p} \cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})} \cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}.
\end{aligned}
$$

Now we bound the initialization time.

$$
\begin{aligned}
\tau_I^3(s,p,q) \;\leq\; & \mathcal{O}\Big(\big(\sum_{\hat{p}\leq s}\tau_I^2(n,\hat{p},s-\hat{p})\big)+\zeta^3(n,p,q)\cdot n^{\mathcal{O}(1)}\Big)\\[4pt]
\leq\; & s2^{s^{s+2}}+2^{\mathcal{O}(s)}\cdot n\log n+\binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot n^{\mathcal{O}(1)}\\[4pt]
\leq\; & 2^{(\log(p+q))^{(\log(p+q))^2+3}}+\binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot n^{\mathcal{O}(1)}.
\end{aligned}
$$

For the degree, we get the following bound.

$$
\begin{aligned}
\Delta^3(n,p,q) \;\leq\; & |\mathscr{P}_t|\cdot\max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p}\prod_{i\leq t}\Delta^2(n,p_i,s-p_i)\\[4pt]
\leq\; & 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p}\prod_{i\leq t}\Big[\big(\frac{s}{s-p_i}\big)^{s-p_i}\cdot s^{\mathcal{O}(1)}\cdot\log n\Big]\\[4pt]
\leq\; & 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot\max_{(q_1,\ldots,q_t)\in\mathcal{Z}_{s,t}^{st-p}}\prod_{i\leq t}\big(\frac{s}{q_i}\big)^{q_i}\\[4pt]
\leq\; & 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot(st)^q\max_{(q_1,\ldots,q_t)\in\mathcal{Z}_{s,t}^q}\frac{1}{\prod_{i\leq t}(q_i\cdot t)^{q_i}}\\[4pt]
\leq\; & 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot(p+q)^q\cdot\frac{1}{q^q}\\[4pt]
\leq\; & \big(\frac{p+q}{q}\big)^q\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}
\end{aligned}
$$

In the second to last transition, the inequality

$$
\max_{(q_1,\ldots,q_t)\in\mathcal{Z}_{s,t}^q}\frac{1}{\prod_{i\leq t}(q_i\cdot t)^{q_i}}\leq\frac{1}{q^q}
$$

follows from Gibbs' inequality [23, Lemma 22.2]. We now consider the bound for the query time.

$$
\begin{aligned}
\tau_Q^3(n,p,q)\;\leq\; & \mathcal{O}\big(\Delta^3(n,p,q)\cdot n^{\mathcal{O}(1)}+|\mathscr{P}_t|\cdot t\cdot\sum_{\hat{p}\leq s}\tau_Q^2(n,\hat{p},s-\hat{p})\big)\\[4pt]
\leq\; & \big(\frac{p+q}{q}\big)^q\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot n^{\mathcal{O}(1)}
\end{aligned}
$$

We now apply Lemma 4.4 to construction number 3, and obtain construction number 4, with the following bounds.

- size $\zeta^4(n,p,q)\leq\binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n$,

- initialization time $\tau_I^4(n,p,q)\leq 2^{(\log(p+q))^{(\log(p+q))^2+3}}+\binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot n\log n$,

- degree $\Delta^4(n,p,q)\leq\big(\frac{p+q}{q}\big)^q\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n$, and

- query time $\tau_Q^4(n,p,q)\leq\big(\frac{p+q}{q}\big)^q\cdot 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})}\cdot\log n$.

To get construction number 5 from construction number 4 we apply Lemma 4.5 again. Just as in the analysis of construction number 3, we set $s=\lfloor(\log(p+q))^2\rfloor$ and $t=\lceil\frac{p+q}{s}\rceil$. The size

of the obtained construction can be bounded as follows.

$$
\begin{aligned}
\zeta^5(n,p,q) &\leq |\mathscr{P}_t| \cdot \sum_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \prod_{i\leq t} \zeta^4(n,p_i,s-p_i) \\
&\leq 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})} \cdot \max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \prod_{i\leq t} \binom{s}{p_i} \cdot 2^{\mathcal{O}(\frac{s}{\log s})} \cdot \log n \\
&\leq 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})} \cdot \max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \prod_{i\leq t} \binom{s}{p_i} \\
&\leq \binom{p+q}{p} \cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}.
\end{aligned}
$$

Next we consider the initialization time.

$$
\begin{aligned}
\tau_I^5(s,p,q) &\leq \mathcal{O}\Big(\big(\sum_{\hat{p}\leq s} \tau_I^4(n,\hat{p},s-\hat{p})\big) + \zeta^5(n,p,q)\cdot n^{\mathcal{O}(1)}\Big) \\
&\leq s2^{(\log s)^{(\log s)^2+3}} + 2^{\mathcal{O}(s)}\cdot n\log n + \binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot n^{\mathcal{O}(1)} \\
&\leq \binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot n^{\mathcal{O}(1)}
\end{aligned}
$$

In the last transition we insert $\log^2(p+q)$ for s and observe that

$$
s2^{(\log s)^{(\log s)^2+3}} \leq 2^{(2\log\log(p+q))^{(2\log\log(p+q))^2+4}} \leq 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}.
$$

We proceed to bound the degree.

$$
\begin{aligned}
\Delta^5(n,p,q) &\leq |\mathscr{P}_t| \cdot \max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \prod_{i\leq t} \Delta^4(n,p_i,s-p_i) \\
&\leq 2^{\mathcal{O}(\frac{p+q}{\log(p+q)})} \cdot \max_{(p_1,\ldots,p_t)\in\mathcal{Z}_{s,t}^p} \prod_{i\leq t} \Big[\big(\frac{s}{s-p_i}\big)^{s-p_i}\cdot 2^{\mathcal{O}(\frac{s}{\log s})}\cdot \log n\Big] \\
&\leq 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})} \cdot \max_{(q_1,\ldots,q_t)\in\mathcal{Z}_{s,t}^{st-p}} \prod_{i\leq t} \big(\frac{s}{q_i}\big)^{q_i} \\
&\leq \big(\frac{p+q}{q}\big)^q \cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}
\end{aligned}
$$

Here the last transition follows from an analysis identical to the last three transitions in the bound for Δ^3. We now consider the query time.

$$
\begin{aligned}
\tau_Q^5(n,p,q) &\leq \mathcal{O}\big(\Delta^5(n,p,q)\cdot n^{\mathcal{O}(1)} + |\mathscr{P}_t|\cdot t\cdot \sum_{\hat{p}\leq s}\tau_Q^4(n,\hat{p},s-\hat{p})\big) \\
&\leq \big(\frac{p+q}{q}\big)^q \cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \log n^{\mathcal{O}(\frac{p+q}{\log^2(p+q)})}\cdot n^{\mathcal{O}(1)}
\end{aligned}
$$

Finally, we apply Lemma 4.4 to construction number 5, and obtain construction number 6, with the following bounds.

- size $\zeta^6(n,p,q) \leq \binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot \log n$,
- initialization time $\tau_I^6(n,p,q) \leq \binom{p+q}{p}\cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})}\cdot n\log n$,

- degree $\Delta^6(n,p,q) \leq (\frac{p+q}{q})^q \cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \log n$, and

- query time $\tau_Q^6(n,p,q) \leq (\frac{p+q}{q})^q \cdot 2^{\mathcal{O}(\frac{p+q}{\log\log(p+q)})} \cdot \log n$.

Construction 6 has the parameters claimed in the statement of the lemma. This concludes the proof. □

5 Applications

In this section we demonstrate how the efficient construction of representative families can be used to design single-exponential parameterized and exact exponential time algorithms. Our applications include best known deterministic algorithms for LONGEST DIRECTED CYCLE, MINIMUM EQUIVALENT GRAPH, k-PATH and k-TREE. We also provide alternate deterministic algorithms running in time $2^{\mathcal{O}(t)}n$ for "connectivity" problems such as HAMILTONIAN CYCLE or STEINER TREE on n-vertex graphs of treewidth at most t.

Let $M = (E, \mathcal{I})$ be a matroid with the ground set of size n and $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a p-family of independent sets. Then for specific matroids we use the following notations to denote the time required to compute the following q-representative families of $\mathcal{S}$:

- $\mathcal{T}_{rm}(t,p,q)$ is the time required to compute a family $\widehat{\mathcal{S}} \subseteq_{rep}^q \mathcal{S}$ of size $\binom{p+q}{q}$, when M is a linear matroid.

- $\mathcal{T}_{um}(t,p,q)$ is the time required to compute a family $\widehat{\mathcal{S}} \subseteq_{rep}^q \mathcal{S}$ of size $\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$, when M is a uniform matroid.

Let us remind, that by Theorem 1, when rank of M is $p+q$, $\mathcal{T}_{rm}(t,p,q)$ is bounded by $\mathcal{O}\left(\binom{p+q}{p}tp^\omega + t\binom{p+q}{q}^{\omega-1}\right)$ multiplied by the time required to perform operations over $\mathbb{F}$. By Theorem 2, $\mathcal{T}_{um}(t,p,q) = \mathcal{O}(t \cdot (\frac{p+q}{q})^q \cdot \log n)$

5.1 Long Directed Cycle

In this section we give our first application of algorithms based on representative families. We study the following problem.

LONGEST DIRECTED CYCLE	**Parameter:** k
Input: A n-vertex and m-arc directed graph D and a positive integer k.	
Question: Does there exist a directed cycle of length at least k in D?	

Observe that the LONGEST DIRECTED CYCLE problem is different from the well-known problem of finding a directed cycle of length *exactly* k. It is quite possible that the only directed cycle that has length at least k is much longer than k, and possibly even is a Hamiltonian cycle. Let D be a directed graph, k be a positive integer, and $M = (E, \mathcal{I})$ be a uniform matroid $U_{n,2k}$ where $E = V(D)$ and $\mathcal{I} = \{S \subseteq V(D) \mid |S| \leq 2k\}$. In this subsection whenever we talk about independent sets, these are independent sets of the uniform matroid $U_{n,2k}$. For a pair of vertices $u, v \in V(D)$, we define

$$\mathcal{P}_{uv}^i = \left\{ X \mid X \subseteq V(D),\ u,v \in X,\ |X| = i,\ \text{and there is a directed } uv\text{-path in } D \right.$$

$$\left. \text{of length } i - 1 \text{ with all the vertices belonging to } X. \right\}$$

We start with a structural lemma providing the key insight to our algorithm.
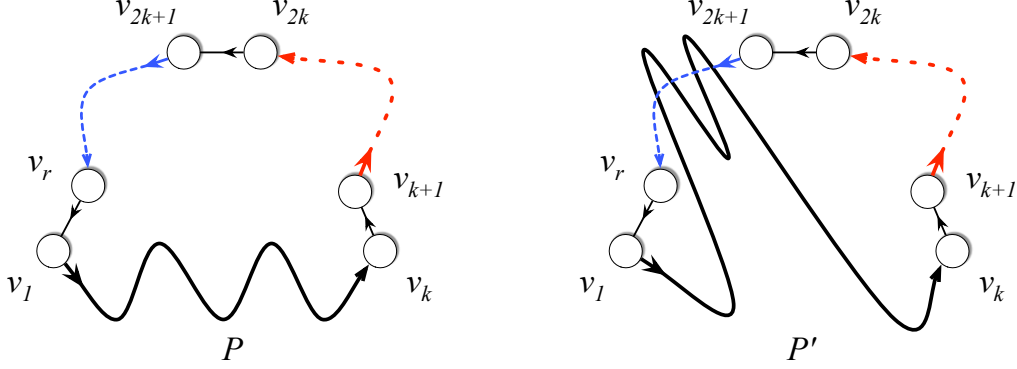
Figure 1: Illustration to the proof of Lemma 5.1.

Lemma 5.1. *Let D be a directed graph. Then D has a directed cycle of length at least k if and only if there exists a pair of vertices $u, v \in V(D)$ and $X \in \widehat{\mathcal{P}}_{uv}^k \subseteq_{rep}^k \mathcal{P}_{uv}^k$ such that D has a directed cycle C and in this cycle vertices of X induce a directed path (that is, vertices of X form a consecutive segment in C).*

Proof. The reverse direction of the proof is straightforward—if cycle C contains a path of length k, the length of C is at least k. We proceed with the proof of the forward direction. Let $C^* = v_1 v_2 \cdots v_r v_1$ be a smallest directed cycle in D of length at least k. That is, $r \geq k$ and there is no directed cycle of length r' where $k \leq r' < r$. We consider two cases.

Case A: $r \leq 2k$. If $r \leq 2k$, then we take $u = v_1$ and $v = v_k$. We define paths $P = v_1 v_2 \cdots v_k$ and $Q = v_{k+1} \cdots v_r$. Because $|Q| \leq k$, by the definition of $\widehat{\mathcal{P}}_{uv}^k \subseteq_{rep}^k \mathcal{P}_{uv}^k$, there exists a directed uv-path P' such that $X = V(P') \in \widehat{\mathcal{P}}_{uv}^k$ and $X \cap Q = \emptyset$. By replacing P with P' in C^* we obtain a directed cycle C of length at least k containing P' as a subpath.

Case B: $r \geq 2k + 1$. In this case we set $u = v_1$, $v = v_k$, and split C^* into three paths $P = v_1 \cdots v_k$, $Q = v_{k+1} \cdots v_{2k}$, and $R = v_{2k+1} \cdots v_r$. Since $|Q| = k$ and $\widehat{\mathcal{P}}_{uv}^k \subseteq_{rep}^k \mathcal{P}_{uv}^k$, it follows that there exists an uv-path P' such that $X = V(P') \in \widehat{\mathcal{P}}_{uv}^k$ and $X \cap Q = \emptyset$. However, P' is not necessarily disjoint with R and by replacing P with P' in C^* we can obtain a *closed walk* C' containing P' as a subpath. See Fig. 1 for an illustration.

If $X \cap R = \emptyset$, then C' is a simple cycle and we take C' as the desired C. We claim that this is the only possibility. Let us assume targeting towards a contradiction that $X \cap R \neq \emptyset$. We want to show that in this case there is a cycle of length at least k but shorter than C^*, contradicting the choice of C^*. Let v_α be the last vertex in $X \cap R$ when we walk from v_1 to v_k along P'. Let $P'[v_\alpha, v_k]$ be the subpath of P starting at v_α and ending at v_k. If $v_\alpha = v_{2k+1}$, we set $R' = \emptyset$. Otherwise we put $R' = R[v_{2k+1}, v_{\alpha-1}]$ to be the subpath of R starting at v_{2k+1} and ending at $v_{\alpha-1}$. Observe that since the arc $v_{\alpha-1} v_\alpha$ is present in D (in fact it is an arc of the cycle C^*), we have that $\overline{C} = P'[v_\alpha, v_k] Q R'$ is a simple cycle in D. Clearly, $|\overline{C}| \geq |Q| \geq k$. Furthermore, since v_1 is not present in $P'[v_\alpha, v_k]$ we have that $|P'[v_\alpha, v_k]| < |P'| = |P|$. Similarly since v_α is not present in R', we have that $|R'| < |R|$. Thus we have

$$k \leq |\overline{C}| = |P'[v_\alpha, v_k]| + |Q| + |R'| < |P| + |Q| + |R| = |C^*|.$$

This implies that $\overline{C}$ is a directed simple cycle of length at least k and strictly smaller than r. This is a contradiction. Hence by replacing P with P' in C^* we obtain a directed cycle C containing P' as a subpath. This concludes the proof. $\qquad\square$

Next lemma provides an efficient computation of family $\widehat{\mathcal{P}}_{uv}^k \subseteq_{rep}^k \mathcal{P}_{uv}^k$.

Lemma 5.2. *Let D be a directed/unidrected graph with n vertices and m edges, $u \in V(D)$ and $M = (E, \mathcal{I})$ be an uniform matroid $U_{n,\ell}$ where $E = V(D)$ and $\mathcal{I} = \{S \subseteq V(D) \mid |S| \leq \ell\}$. Then for every $p \leq \ell$ and $v \in V(D) \setminus \{u\}$, a family $\widehat{\mathcal{P}}_{uv}^p \subseteq_{rep}^{\ell-p} \mathcal{P}_{uv}^p$ of size at most*

$$\binom{\ell}{p} \cdot 2^{o(\ell)} \cdot \log n$$

can be found in time

$$\mathcal{O}\left(2^{o(\ell)} m \log^2 n \max_{i \in [p]} \left\{ \binom{\ell}{i-1} \left(\frac{\ell}{\ell-i}\right)^{\ell-i} \right\}\right).$$

Furthermore, within the same running time every set in $\widehat{\mathcal{P}}_{uv}^p$ can be ordered in a way that it corresponds to a directed (undirected) path in D.

Proof. We prove the lemma only for digraphs. The proof for undirected graphs is analogous and we only point out the differences with the proof for the directed case. We describe a dynamic programming based algorithm. Let $V(D) = \{u, v_1, \ldots, v_{n-1}\}$ and $\mathcal{D}$ be a $(p-1) \times n$ matrix where the rows are indexed from integers in $\{2, \ldots, p\}$ and the columns are indexed from vertices in $\{v_1, \ldots, v_{n-1}\}$. The entry $\mathcal{D}[i, v]$ will store the family $\widehat{\mathcal{P}}_{uv}^i \subseteq_{rep}^{\ell-i} \mathcal{P}_v^i$. We fill the entries in the matrix $\mathcal{D}$ in the increasing order of rows. For $i = 2$, $\mathcal{D}[2, v] = \{\{u, v\}\}$ if $uv \in A(D)$ (for an undirected graph we check whether u and v are adjacent). Assume that we have filled all the entries until the row i. Let

$$\mathcal{N}_{uv}^{i+1} = \bigcup_{w \in N^-(v)} \widehat{\mathcal{P}}_{uw}^i \bullet \{v\}.$$

For undirected graphs we use the following definition

$$\mathcal{N}_{uv}^{i+1} = \bigcup_{w \in N(v)} \widehat{\mathcal{P}}_{uw}^i \bullet \{v\}.$$

Claim 5.1. $\mathcal{N}_{uv}^{i+1} \subseteq_{rep}^{\ell-(i+1)} \mathcal{P}_{uv}^{i+1}$.

Proof. Let $S \in \mathcal{P}_{uv}^{i+1}$ and Y be a set of size $\ell - (i+1)$ (which is essentially an independent set of $U_{n,\ell}$) such that $S \cap Y = \emptyset$. We will show that there exists a set $S' \in \mathcal{N}_{uv}^{i+1}$ such that $S' \cap Y = \emptyset$. This will imply the desired result. Since $S \in \mathcal{P}_{uv}^{i+1}$ there exists a directed path $P = ua_1 \cdots a_{i-1} v$ in D such that $S = \{u, a_1, \ldots, a_{i-1}, v\}$ and $a_i \in N^-(v)$. The existence of path $P[u, a_{i-1}]$, the subpath of P between u and a_{i-1}, implies that $X^* = X \setminus \{v\} \in \mathcal{P}_{ua_{i-1}}^i$. Take $Y^* = Y \cup \{v\}$. Observe that $X^* \cap Y^* = \emptyset$ and $|Y^*| = \ell - i$. Since $\widehat{\mathcal{P}}_{ua_{i-1}}^i \subseteq_{rep}^{\ell-i} \mathcal{P}_{ua_{i-1}}^i$ there exists a set $\widehat{X}^* \in \widehat{\mathcal{P}}_{ua_{i-1}}^i$ such that $\widehat{X}^* \cap Y^* = \emptyset$. However, since $a_{i-1} \in N^-(v)$ and $\widehat{X}^* \cap \{v\} = \emptyset$ (as $\widehat{X}^* \cap Y^* = \emptyset$), we have $\widehat{X}^* \bullet \{v\} = \widehat{X}^* \cup \{v\}$ and $\widehat{X}^* \cup \{v\} \in \mathcal{N}_v^{i+1}$. Taking $S' = \widehat{X}^* \cup \{v\}$ suffices for our purpose. This completes the proof of the lemma. $\square$

We fill the entry for $\mathcal{D}[i+1, v]$ as follows. Observe that

$$\mathcal{N}_{uv}^{i+1} = \bigcup_{w \in N^-(v)} \mathcal{D}[i, w] \bullet \{v\}.$$

We already have computed the family corresponding to $\mathcal{D}[i, w]$ for $w \in N^-(v)$. By Theorem 2, $|\widehat{\mathcal{P}}_{uw}^i| \leq \binom{\ell}{i} 2^{o(\ell)} \log n$ and thus $|\mathcal{N}_{uv}^{i+1}| \leq d^-(v)\binom{\ell}{i} 2^{o(\ell)} \log n$. Furthermore, we can compute

$\mathcal{N}_{uv}^{i+1}$ in time $\mathcal{O}\left(d^-(v)\binom{\ell}{i}2^{o(\ell)}\log n\right)$. Now using Theorem 2, we compute $\widehat{\mathcal{N}}_{uv}^{i+1} \subseteq_{rep}^{\ell-i-1} \mathcal{N}_{uv}^{i+1}$ in time $\mathcal{T}_{um}(t, i+1, \ell-i-1)$, where $t = d(v)\binom{\ell}{i}2^{o(\ell)}\log n$. By Claim 5.1, we know that $\mathcal{N}_{uv}^{i+1} \subseteq_{rep}^{\ell-i-1} \mathcal{P}_{uv}^{i+1}$. Thus Lemma 3.1 implies that $\widehat{\mathcal{N}}_{uv}^{i+1} = \widehat{\mathcal{P}}_{uv}^{i+1} \subseteq_{rep}^{\ell-i-1} \mathcal{P}_{uv}^{i+1}$. We assign this family to $\mathcal{D}[i+1, v]$. This completes the description and the correctness of the algorithm. We give ordering to the vertices of the sets in $\widehat{\mathcal{P}}_{uv}^p$ in the following way so that it corresponds to a directed (undirected) path in D. We keep the sets in the order in which they are built using the $\bullet$ operation. That is, we can view these sets as strings and $\bullet$ operation as concatenation. Then every ordered set in our family represents a path in the graph. The running time of the algorithm is bounded by

$$\mathcal{O}\left(\sum_{i=2}^{p}\sum_{j=1}^{n-1}\mathcal{T}_{um}\left(d^-(v_j)\binom{\ell}{i-1}2^{o(\ell)}\log n, i, \ell-i\right)\right)$$

$$= \mathcal{O}\left(\sum_{i=2}^{p}\sum_{j=1}^{n-1}d^-(v_j)\binom{\ell}{i-1}\left(\frac{\ell}{\ell-i}\right)^{\ell-i}2^{o(\ell)}\log^2 n\right)$$

$$= \mathcal{O}\left(2^{o(\ell)}\log^2 n\sum_{i=2}^{p}\sum_{j=1}^{n-1}d^-(v_j)\binom{\ell}{i-1}\left(\frac{\ell}{\ell-i}\right)^{\ell-i}\right)$$

$$= \mathcal{O}\left(2^{o(\ell)}\log^2 nm\max_{i\in[p]}\left\{\binom{\ell}{i-1}\left(\frac{\ell}{\ell-i}\right)^{\ell-i}\right\}\right)$$

This completes the proof. $\qquad\square$

Finally, we are ready to state the main result of this section.

Theorem 6. LONGEST DIRECTED CYCLE *can be solved in time* $\mathcal{O}(8^{k+o(k)}mn^2\log n)$.

Proof. Let D be a directed graph. We solve the problem by applying the structural characterization proved in Lemma 5.1. By Lemma 5.1, D has a directed cycle of length at least k if and only if there exists a pair of vertices $u, v \in V(D)$ and a path P' with $V(P') \in \widehat{\mathcal{P}}_{uv}^k \subseteq_{rep}^k \mathcal{P}_{uv}^k$ such that D has a directed cycle C containing P' as a subpath.

We first compute $\widehat{\mathcal{P}}_{uv}^k \subseteq_{rep}^k \mathcal{P}_{uv}^k$ for all $u, v \in V(D)$. For that we apply Lemma 5.2 for each vertex $u \in V(D)$ with $\ell = 2k$ and $p = k$. Thus, we can compute $\widehat{\mathcal{P}}_{uv}^k \subseteq_{rep}^k \mathcal{P}_{uv}^k$ for all $u, v \in V(D)$ in time $\mathcal{O}\left(8^{k+o(k)}mn\log^2 n\right)$. Moreover, for every $X \in \widehat{\mathcal{P}}_{uv}^k$ we also compute a directed uv-path P_X using vertices of X. Let

$$\mathcal{Q} = \bigcup_{u,v\in V(D)}\widehat{\mathcal{P}}_{uv}^k.$$

Now for every set $X \in \mathcal{Q}$ and the corresponding uv-path P_X with endpoint, we check if there is a uv-path in D avoiding all vertices of X but u and v. This check can be done by a standard graph traversal algorithm like BFS/DFS in time $\mathcal{O}(m+n)$. If we succeed in finding a path for at least one $X \in \mathcal{Q}$, we answer YES and return the corresponding directed cycle obtained by merging P_X and another path. Otherwise, if we did not succeed to find such a path for none of the sets $X\mathcal{Q}$, this means that there is no directed cycle of length at least k in D. The correctness of the algorithm follows from Lemma 5.1. By Theorem 2, the size of $\mathcal{Q}$ is upper bounded by $n^2\binom{2k}{k}2^{o(k)}\log n \le n^2 4^{k+o(k)}\log n$. Thus the overall running time of the algorithm is upper bounded by

$$\mathcal{O}(8^{k+o(k)}mn\log^2 n + 4^{k+o(k)}(n^2 m + n^3)\log n).$$

This concludes the proof. $\qquad\square$

27

5.2 Minimum Equivalent Graph

For a given digraph D, a subdigraph D' of D is said to be an *equivalent* subdigraph of D if for any pair of vertices $u, v \in V(D)$ if there is a directed path in D from u to v then there is also a directed path from u to v in D'. That is, reachability of vertices in D and D' is same. In this section we study a problem where given a digraph D the objective is to find an equivalent subdigraph of D' of D with as few arcs as possible. Equivalently, the objective is to remove the maximum number of arcs from a digraph D without affecting its reachability. More precisely the problem we study is as follows.

MINIMUM EQUIVALENT GRAPH (MEG)
Input: A directed graph D
Task: Find an equivalent subdigraph of D with the minimum number of arcs.

The following proposition is due to Moyles and Thompson [36], see also [3, Sections 2.3], reduces the problem of finding a minimum equivalent subdigraph of an arbitrary D to a strong digraph.

Proposition 5.1. *Let D be a digraph on n vertices with strongly connected components $C_1, \ldots, C_r$. Given a minimum equivalent subdigraph C_i' for each C_i, $i \in [r]$, one can obtain a minimum equivalent subdigraph D' of D containing each of C_i' in $\mathcal{O}(n^\omega)$ time.*

Observe that for a strong digraph D any equivalent subdigraph is also strong. By Proposition 5.1, MEG reduces to the following problem.

MINIMUM STRONGLY CONNECTED SPANNING SUBGRAPH (MINIMUM SCSS)
Input: A strongly connected directed graph D
Task: Find a strong spanning subdigraph of D with the minimum number of arcs.

It seems to be no established agreement in the literature on how to call these problems. MEG sometimes is also referred as MINIMUM EQUIVALENT DIGRAPH and MINIMUM EQUIVALENT SUBDIGRAPH, while MINIMUM SCSS is also called MINIMUM SPANNING STRONG SUBDIGRAPH (MSSS).

A digraph T is an *out-tree* (an *in-tree*) if T is an oriented tree with just one vertex s of in-degree zero (out-degree zero). The vertex s is the root of T. If an out-tree (in-tree) T is a spanning subdigraph of D, T is called an *out-branching* (an *in-branching*). We use the notation B_s^+ (B_s^+) to denote an out-branching (in-branching) rooted at s of the digraph.

It is known that a digraph is strong if and only if it contain an out-branching and an in-branching rooted at some vertex $v \in V(D)$ [3, Proposition 12.1.1].

Proposition 5.2. *Let D be a strong digraph on n vertices, let v be an arbitrary vertex of $V(D)$, and $\ell \leq n - 2$ be a natural number. Then there exists a strong spanning subdigraph of D with at most $2n - 2 - \ell$ arcs if and only if D contains an in-branching B_v^- and an out-branching B_v^+ with root v so that $|A(B_v^+) \cap A(B_v^-)| \geq \ell$ (that is, they have at least ℓ common arcs).*

Proposition 5.2 implies that the MINIMUM SCSS problem is equivalent to finding, for an arbitrary vertex $v \in V(D)$, an out-branching B_v^+ and an in-branching B_v^- that maximizes $|A(B_v^+) \cap A(B_v^-)|$. For our exact algorithm for MINIMUM SCSS we implement this equivalent version using representative sets.

Let D be a strong digraph and $s \in V(D)$ be a fixed vertex. For $v \in V(D)$ we use $\mathsf{In}(v)$ and $\mathsf{Out}(v)$ to denote the sets of in-coming and out-coming arcs incident with v. By D_s^- we denote

the digraph obtained from D by deleting the arcs in $\mathsf{Out}(s)$. Similarly, by D_s^+ we denote the digraph obtained from D by deleting the arcs in $\mathsf{In}(s)$.

We take two copies E_1, E_2 of $A(D)$ (that is $E_i = \{e_i \mid e \in A(D)\}$) , a copy E_3 of $A(D_s^+)$ and a copy E_4 of $A(D_s^-)$ and construct four matroids as follows. Let $U(D)$ denote the underlying undirected graph of D. The first two matroids $M_1 = (E_1, \mathcal{I}_1)$, $M_2 = (E_2, \mathcal{I}_2)$ are the graphic matroids on $U(D)$. Observe that

$$A(D_s^+) = \biguplus_{v \in V(D_s^+)} \mathsf{In}(v) \text{ and } A(D_s^-) = \biguplus_{v \in V(D_s^-)} \mathsf{Out}(v).$$

Thus the arcs of D_s^+ can be partitioned into sets of in-arcs and similarly the arcs of D_s^- into sets of out-arcs. The other two matroids are the following partition matroids $M_3 = (E_3, \mathcal{I}_3)$, $M_4 = (E_4, \mathcal{I}_4)$, where

$$\mathcal{I}_3 = \{I \mid I \subseteq A(D_s^+), \text{ for every } v \in V(D_s^+) = V(D), |I \cap \mathsf{In}(v)| \leq 1\},$$

and

$$\mathcal{I}_4 = \{I \mid I \subseteq A(D_s^-), \text{ for every } v \in V(D_s^-) = V(D), |I \cap \mathsf{Out}(v)| \leq 1\}.$$

We define the matroid $M = (E, \mathcal{I})$ as the direct sum $M = M_1 \oplus M_2 \oplus M_3 \oplus M_4$. Since each of M_i is a representable matroids over the same field (by Propositions 2.2 and 2.3), we have that M is also representable (Proposition 2.1). The reason we say that M_i is representable over the same field $\mathbb{F}$ is that the graphic matroid is representable over any field and the partition matroids defined here are representable over a finite field of size $n^{\mathcal{O}(n)}$. So if we take $\mathbb{F}$ as a finite field of size $n^{\mathcal{O}(n)}$ then M is representable over $\mathbb{F}$. The rank of this matroid is $4n - 4$.

Let us note that for each arc $e \in A(D)$ which is not incident with s, we have four elements in the matroid M, corresponding to the copies of e in M_i, $i \in \{1, \ldots, 4\}$. We denote these elements by e_i, $i \in \{1, \ldots, 4\}$. For every edge $e \in A(D)$ incident with s, we have three corresponding elements. We denote them by e_1, e_2, e_3, or e_1, e_2, e_4, depending on the case when e is in- or out-arc for s.

For $i \in \{1, \ldots, n-1\}$, we define

$$\mathcal{B}^{4i} = \Big\{ W \mid W \in \mathcal{I}, \ |W| = 4i, \ \forall \, e \in A(D) \text{ either } W \cap \{e_1, e_2, e_3, e_4\} = \emptyset \text{ or } \{e_1, e_2, e_3, e_4\} \subseteq W \Big\}.$$

For $W \in \mathcal{I}$, by A_W we denote the set of arcs $e \in A(D)$ such that $\{e_1, e_2, e_3, e_4\} \cap W \neq \emptyset$. Now we are ready to state the lemma that relates representative sets and the Minimum SCSS problem.

Lemma 5.3. *Let D be a strong digraph on n vertices and $\ell \leq n - 2$ be a natural number. Then there exists a strong spanning subdigraph D' of D with at most $2n - 2 - \ell$ arcs if and only if there exists a set $\widehat{F} \in \widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ such that D has a strong spanning subdigraph $\bar{D}$ with $A_{\widehat{F}} \subseteq A(\bar{D})$. Here, $n' = 4n - 4$.*

Proof. We only show the forward direction of the proof, the reverse direction is straightforward. Let D' be a strong spanning subdigraph of D with at most $2n - 2 - \ell$ arcs. Thus, by Proposition 5.2 we have that for any vertex $v \in V(D')$, there exists an out-branching B_v^+ and an in-branching B_v^- in D' such that $|A(B_v^+) \cap A(B_v^-)| \geq \ell$. Observe that the arcs in $A(B_v^+) \cap A(B_v^-)$ form an out-forest (in-forest). Let F' be an arbitrary subset of $A(B_v^+) \cap A(B_v^-)$ containing exactly ℓ arcs. Take $X = A(B_v^+) \setminus F'$ and $Y = A(B_v^-) \setminus F'$. Observe that X and Y need not be disjoint. Clearly, $|X| = |Y| = n - 1 - \ell$.

In the matroid M, one can associate with D' an independent set $I_{D'}$ of size $4n-4$ as follows:

$$I_{D'} = \bigcup_{e \in F'} \{e_1, e_2, e_3, e_4\} \bigcup_{e \in X} \{e_1, e_3\} \bigcup_{e \in Y} \{e_2, e_4\}.$$

By our construction, we have that $I_{D'}$ is an independent set in $\mathcal{I}$ and $|I_{D'}| = 4\ell + 4(n-1-\ell) = n'$. Let $F = \bigcup_{e \in F'} \{e_1, e_2, e_3, e_4\}$, $\bar{X} = \bigcup_{e \in X} \{e_1, e_3\}$ and $\bar{Y} = \bigcup_{e \in Y} \{e_2, e_4\}$. Then notice that $F \in \mathcal{B}^{4\ell}$ and $F \subset I_{D'}$. This implies that there exists a set $\widehat{F} \in \widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ such that $I_{\bar{D}} = \widehat{F} \cup \bar{X} \cup \bar{Y} \in \mathcal{I}$. We show that D has a strong spanning subdigraph $\bar{D}$ with $A_{\widehat{F}} \subseteq A(\bar{D})$. Let $\bar{D}$ be the digraph with the vertex set $V(D)$ and the arc set $A(\bar{D}) = X \cup Y \cup A_{\widehat{F}}$. Consider the following four sets.

1. Let $W_1 = \{e_1 \mid e \in X \cup A_{\widehat{F}}\}$ then we have that $W_1 \subseteq I_{\bar{D}}$ and thus $W_1 \in \mathcal{I}_1$. This together with the fact that $|W_1| = n-1$ implies that $X \cup A_{\widehat{F}}$ forms a spanning tree in $U(D)$.

2. Let $W_2 = \{e_2 \mid e \in Y \cup A_{\widehat{F}}\}$. Similar to the first case, then $Y \cup A_{\widehat{F}}$ forms a spanning tree in $U(D)$.

3. Let $W_3 = \{e_3 \mid e \in X \cup A_{\widehat{F}}\}$ then we have that $W_3 \subseteq I_{\bar{D}}$ and thus $W_3 \in \mathcal{I}_3$. This together with the fact that $|W_1| = |W_3| = n-1$ and that $X \cup A_{\widehat{F}}$ is a a spanning tree in $U(D)$ implies that $X \cup A_{\widehat{F}}$ forms an out-branching rooted at s in D_s^+.

4. Let $W_4 = \{e_3 \mid e \in Y \cup A_{\widehat{F}}\}$. Similar to the previous case, then $Y \cup A_{\widehat{F}}$ forms an in-branching rooted at s in D_s^-.

We have shown that $\bar{D}$ contains $A_{\widehat{F}}$ and has an out-branching and in-branching rooted at s. This implies that $\bar{D}$ is the desired strong spanning subdigraph of D containing a set from $\widehat{\mathcal{B}}^{4\ell}$. This concludes the proof of the lemma. $\qquad\square$

Lemma 5.4. *Let D be a strong digraph on n vertices and $\ell \leq n-2$ be a natural number. Then in time $\mathcal{O}\left(\max_{i \in [\ell]} \binom{n'}{4i}^{\omega} mn^2 \log n\right)$ we can compute $\widehat{\mathcal{B}}^{4\ell} \subseteq_{rep}^{n'-4\ell} \mathcal{B}^{4\ell}$ of size $\binom{n'}{4\ell}$. Here, $n' = 4n-4$.*

Proof. We describe a dynamic programming based algorithm. Let $\mathcal{D}$ be an array of size ℓ. The entry $\mathcal{D}[i]$ will store the family $\widehat{\mathcal{B}}^{4i} \subseteq_{rep}^{n'-4i} \mathcal{B}^{4\ell}$. We fill the entries in the array $\mathcal{D}$ in the increasing order of its index, that is, from $0, \ldots, \ell$. For the base case define $\widehat{\mathcal{B}}^0 = \{\emptyset\}$ and let $W = \{\{e_1, e_2, e_3, e_4\} \mid e \in A(D)\}$. Given that $\mathcal{D}[i]$ is filled for all $i' \leq i$, we fill $\mathcal{D}[i+1]$ as follows. Define $\mathcal{N}^{4(i+1)} = \left(\widehat{\mathcal{B}}^{4i} \bullet W\right) \cap \mathcal{I}$.

Claim 5.2. *For all $0 \leq i \leq \ell - 1$, $\mathcal{N}^{4(i+1)} \subseteq_{rep}^{n'-4(i+1)} \mathcal{B}^{4(i+1)}$.*

Proof. Let $S \in \mathcal{B}^{4(i+1)}$ and Y be a set of size $n' - 4(i+1)$ such that $S \cap Y = \emptyset$ and $S \cup Y \in \mathcal{I}$. We will show that there exists a set $\widehat{S} \in \mathcal{N}^{4(i+1)}$ such that $\widehat{S} \cap Y = \emptyset$ and $\widehat{S} \cup Y \in \mathcal{I}$. This will imply the desired result.

Let $e \in A(D)$ such that $\{e_1, e_2, e_3, e_4\} \subseteq S$. Define $S^* = S \setminus \{e_1, e_2, e_3, e_4\}$ and $Y^* = Y \cup \{e_1, e_2, e_3, e_4\}$. Since $S \cup Y \in \mathcal{I}$ we have that $S^* \in \mathcal{I}$ and $Y^* \in \mathcal{I}$. Observe that $S^* \in \mathcal{B}^{4i}$, $S^* \cup Y^* \in \mathcal{I}$ and the size of Y^* is $n' - 4i$. This implies that there exists $\widehat{S}^*$ in $\widehat{\mathcal{B}}^{4i} \subseteq_{rep}^{n'-4i} \mathcal{B}^{4\ell}$ such that $\widehat{S}^* \cup Y^* \in \mathcal{I}$. Thus $\widehat{S}^* \cup \{e_1, e_2, e_3, e_4\} \in \mathcal{I}$ and also in $\widehat{\mathcal{B}}^{4i} \bullet W$ and thus in $\mathcal{N}^{4(i+1)}$. Taking $\widehat{S} = \widehat{S}^* \cup \{e_1, e_2, e_3, e_4\}$ suffices for our purpose. This completes the proof of the claim. $\qquad\square$

We fill the entry for $\mathcal{D}[i+1]$ as follows. Observe that $\mathcal{N}^{i+1}_{uv} = (\mathcal{D}[i,w] \bullet W) \cap \mathcal{I}$. We already have computed the family corresponding to $\mathcal{D}[i]$. By Theorem 1, $|\widehat{\mathcal{B}}^{4i}| \le \binom{n'}{4i}$ and thus $|\mathcal{N}^{4(i+1)}| \le 4m\binom{n'}{4i}$. Furthermore, we can compute $\mathcal{N}^{4(i+1)}$ in time $\mathcal{O}\left(mn\binom{n'}{4i}\right)$. Now using Theorem 1, we can compute $\widehat{\mathcal{N}}^{4(i+1)} \subseteq^{n'-4(i+1)}_{rep} \mathcal{N}^{4(i+1)}$ in time $\mathcal{T}_{rm}(t, 4i+4, n'-4(i+1))$, where $t = 4m\binom{n'}{4i}$.

By Claim 5.2 we know that $\mathcal{N}^{4(i+1)} \subseteq^{n'-4(i+1)}_{rep} \mathcal{B}^{4(i+1)}$. Thus Lemma 3.1 implies that $\widehat{\mathcal{N}}^{4(i+1)} = \widehat{\mathcal{B}}^{4(i+1)} \subseteq^{n'-4(i+1)}_{rep} \mathcal{B}^{4(i+1)}$. We assign this family to $\mathcal{D}[i+1]$. This completes the description and the correctness of the dynamic programming. The field size for uniform matroids are upper bounded by $n^{\mathcal{O}(n^2)}$ and thus we can perform all the field operations in time $\mathcal{O}(n^2 \log n)$. Thus, the running time of this algorithm is upper bounded by

$$\mathcal{O}\left(\sum_{i=1}^{\ell} \mathcal{T}_{rm}\left(4m\binom{n'}{4(i-1)}, 4i, n'-4i\right)\right) = \mathcal{O}\left(\max_{i \in [\ell]} \binom{n'}{4i}^{\omega} mn^2 \log n\right).$$

This completes the proof. $\qquad\square$

Lemma 5.5. MINIMUM SCSS *can be solved in time* $\mathcal{O}(2^{4\omega n} m^2 n)$.

Proof. Let us fix $n' = 4n - 4$. Proposition 5.2 implies that the MINIMUM SCSS problem is equivalent to finding, for an arbitrary vertex $s \in V(D)$, an out-branching B_v^+ and an in-branching B_v^- that maximizes $|A(B_v^+) \cap A(B_v^-)|$. We guess the value of $|A(B_v^+) \cap A(B_v^-)|$ and let this be ℓ. By Lemma 5.3, there exists a strong spanning subdigraph D' of D with at most $2n - 2 - \ell$ arcs if and only if there exists a set $\widehat{F} \in \widehat{\mathcal{B}}^{4\ell} \subseteq^{n'-4\ell}_{rep} \mathcal{B}^{4\ell}$ such that D has a strong spanning subdigraph $\bar{D}$ with $A_{\widehat{F}} \subseteq A(\bar{D})$. Recall that for $X \in \mathcal{I}$, by A_X we denote the set of arcs $e \in A(D)$ such that $\{e_1, e_2, e_3, e_4\} \cap X \ne \emptyset$. Now using Lemma 5.4 we compute $\widehat{\mathcal{B}}^{4\ell} \subseteq^{n'-4\ell}_{rep} \mathcal{B}^{4\ell}$ in time $\mathcal{O}\left(\max_{i \in [\ell]} \binom{n'}{4i}^{\omega} mn^2 \log n\right)$.

For every $\widehat{F} \in \widehat{\mathcal{B}}^{4\ell}$ we test whether $A_{\widehat{F}}$ can be extended to an out-branching in D_s^+ and to an in-branching in D_s^-. We can do it in $\mathcal{O}(n(n+m))$-time by putting weights 0 to the arcs of $A_{\widehat{F}}$ and weights 1 to all remaining arcs and then by running the classical algorithm of Edmonds [16]. Since $\ell \le n - 2$, the running time of this algorithm is upper bounded by $\mathcal{O}(2^{4\omega n} m^2 n)$. This concludes the proof. $\qquad\square$

Finally, we are ready to prove the main result of this section

Theorem 7. MINIMUM EQUIVALENT GRAPH *can be solved in time* $\mathcal{O}(2^{4\omega n} m^2 n)$.

Proof. Given an arbitrary digraph D we first find its strongly connected components $C_1, \ldots, C_s$. Now on each C_i, we apply Lemma 5.5 and obtain a minimum equivalent subdigraph C_i'. After this we apply Proposition 5.1 and obtain a minimum equivalent subdigraph of D. Since all the steps except Lemma 5.5 takes polynomial time we get the desired running time. This completes the proof. $\qquad\square$

A weighted variant of MINIMUM EQUIVALENT GRAPH has also been studied in literature. More precisely the problem is defined as follows.

MINIMUM WEIGHT EQUIVALENT GRAPH (MWEG)
Input: A directed graph D and a weight function $w : A(D) \to \mathbb{N}$.
Task: Find a minimum weight equivalent subdigraph of D.

MWEG can be solved along the same line as MEG but to do this we need to use the notion of min q-representative family and use Theorem 3 instead of Theorem 1. These changes give us the following theorem.

Theorem 8. MINIMUM WEIGHT EQUIVALENT GRAPH *can be solved in time* $\mathcal{O}(2^{4\omega n}m^2n)$.

5.3 Dynamic Programming over graphs of bounded treewidth

In this section we discuss deterministic algorithms for "connectivity problems" such as HAMILTONIAN PATH, STEINER TREE, FEEDBACK VERTEX SET parameterized by the treewidth of the input graph. The algorithms are based on Theorem 1 and use graphic matroids to take care of connectivity constraints. The approach is generic and can be used whenever all the relevant information about a "partial solution" can be encoded as an independent set of a specific linear matroid. We exemplify the approach on the STEINER TREE problem.

STEINER TREE
Input: An undirected graph G with a set of terminals $T \subseteq V(G)$, and a weight
    function $w : E(G) \to \mathbb{N}$.
Task: Find a subtree in G of minimum weight spanning all vertices of T.

5.3.1 Treewidth

Let G be a graph. A *tree-decomposition* of a graph G is a pair $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ such that

- $\cup_{t \in V(\mathbb{T})} X_t = V(G)$,
- for every edge $xy \in E(G)$ there is a $t \in V(\mathbb{T})$ such that $\{x, y\} \subseteq X_t$, and
- for every vertex $v \in V(G)$ the subgraph of $\mathbb{T}$ induced by the set $\{t \mid v \in X_t\}$ is connected.

The *width* of a tree decomposition is $\max_{t \in V(\mathbb{T})} |X_t| - 1$ and the *treewidth* of G is the minimum width over all tree decompositions of G and is denoted by $\mathbf{tw}(G)$.

A tree decomposition $(\mathbb{T}, \mathcal{X})$ is called a *nice tree decomposition* if $\mathbb{T}$ is a tree rooted at some node r where $X_r = \emptyset$, each node of $\mathbb{T}$ has at most two children, and each node is of one of the following kinds:

1. **Introduce node**: a node t that has only one child t' where $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$.

2. **Forget node**: a node t that has only one child t' where $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$.

3. **Join node**: a node t with two children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$.

4. **Base node**: a node t that is a leaf of $\mathbb{T}$, is different than the root, and $X_t = \emptyset$.

Notice that, according to the above definition, the root r of $\mathbb{T}$ is either a forget node or a join node. It is well known that any tree decomposition of G can be transformed into a nice tree decomposition maintaining the same width in linear time [24]. We use G_t to denote the graph induced by the vertex set $\cup_{t'} X_{t'}$, where t' ranges over all descendants of t, including t. By $E(X_t)$ we denote the edges present in $G[X_t]$. We use H_t to denote the graph on vertex set $V(G_t)$ and the edge set $E(G_t) \setminus E(X_t)$. For clarity of presentation we use the term nodes to refer to the vertices of the tree $\mathbb{T}$.

5.3.2 STEINER TREE **parameterized by treewidth**

Let G be an input graph of the STEINER TREE problem. Throughout this section, we say that $E' \subseteq E(G)$ is a *solution* if the subgraph induced on this edge set is connected and it contains all the terminal vertices. We call $E' \subseteq E(G)$ an *optimal solution* if E' is a solution of the minimum weight. Let $\mathscr{S}$ be a family of edge subsets such that every edge subset corresponds to an optimal solution. That is,

$$\mathscr{S} = \{E' \subseteq E(G) \mid E' \text{ is an optimal solution}\}.$$

We start with few definitions that will be useful in explaining the algorithm. Let $(\mathbb{T}, \mathcal{X})$ be a tree decomposition of G of width **tw**. Let t be a node of $V(\mathbb{T})$. By $\mathcal{S}_t$ we denote the family of edge subsets of $E(H_t)$, $\{E' \subseteq E(H_t)\}$, that satisfies the following properties.

- Either E' is a solution (that is, the subgraph induced on this edge set is connected and it contains all the terminal vertices); or

- every vertex of $(T \cap V(G_t)) \setminus X_t$ is incident with some edge from E', and every connected component of the graph induced by E' contains a vertex from X_t.

We call $\mathcal{S}_t$ a *family of partial solutions* foe t. We denote by K^t a complete graph on the vertex set X_t. For an edge subset $E^* \subseteq E(G)$ and bag X_t corresponding to a node t, we define the following.

1. Set $\partial^t(E^*) = X_t \cap V(E^*)$, the set of endpoints of E^* in X_t.

2. Let G^* be the subgraph of G on the vertex set $V(G)$ and the edge set E^*. Let $C'_1, \ldots, C'_\ell$ be the connected components of G^* such that for all $i \in [\ell]$, $C'_i \cap X_t \neq \emptyset$. Let $C_i = C'_i \cap X_t$. Observe that $C_1, \ldots, C_\ell$ is a partition of $\partial^t(E^*)$. By $F(E^*)$ we denote a forest $\{Q_1, \ldots, Q_\ell\}$ where each Q_i is an arbitrary spanning tree of $K^t[C_i]$. For an example, since $K^t[C_i]$ is a complete graph we could take Q_i as a star. The purpose of $F(E^*)$ is to keep track for the vertices in C_i whether they were in the same connected component of G^*.

3. We define $w(F(E^*)) = w(E^*)$.

Our description of the algorithm slightly deviates from the usual table look-up based expositions of dynamic programming algorithms on graphs of bounded treewidth. With every node t of $\mathbb{T}$, we associate a subgraph of G. In our case it will be H_t. For every node t, rather than keeping a table, we keep a family of partial solutions for the graph H_t. That is, for every optimal solution $L \in \mathscr{S}$ and its intersection $L_t = E(H_t) \cap L$ with the graph H_t, we have some partial solution in the family that is "as good as L_t". More precisely, we have some partial solution, say $\hat{L}_t$ in our family such that $\hat{L}_t \cup L_R$ is also an optimum solution for the whole graph. Here, $L_R = L \setminus L_t$. As we move from one node t in the decomposition tree to the next node t' the graph H_t changes to $H_{t'}$, and so does the set of partial solutions. The algorithm updates its set of partial solutions accordingly. Here matroids come into play: in order to bound the size of the family of partial solutions that the algorithm stores at each node we employ Theorem 3 for graphic matroids. More details are given in the proof of the following theorem, which is the main result of this section.

Theorem 9. *Let G be an n-vertex graph given together with its tree decomposition of with **tw**. Then* STEINER TREE *on G can be solved in time* $\mathcal{O}((1 + 2^{\omega+1})^{\mathbf{tw}}\mathbf{tw}^{\mathcal{O}(1)}n)$.

Proof. We first outline an algorithm with running time $\mathcal{O}((1 + 2^{\omega+1})^{\mathbf{tw}}\mathbf{tw}^{\mathcal{O}(1)}n^2)$ for a simple exposition. Later we point out how we can remove the extra factor of n at the cost of a factor polynomial in $\mathbf{tw}$.

For every node t of $\mathbb{T}$ and subset $Z \subseteq X_t$, we store a family of edge subsets $\widehat{\mathcal{S}}_t[Z]$ of H_t satisfying the following correctness invariant.

> **Correctness Invariant:** For every $L \in \mathscr{S}$ we have the following. Let $L_t = E(H_t) \cap L$, $L_R = L \setminus L_t$, and $Z = \partial^t(L)$. Then there exists $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$ such that $w(\hat{L}_t) \leq w(L_t)$, $\hat{L} = \hat{L}_t \cup L_R$ is a solution, and $\partial^t(\hat{L}) = Z$. Observe that since $w(\hat{L}_t) \leq w(L_t)$ and $L \in \mathscr{S}$, we have that $\hat{L} \in \mathscr{S}$.

We process the nodes of the tree $\mathbb{T}$ from base nodes to the root node while doing the dynamic programming. Throughout the process we maintain the correctness invariant, which will prove the correctness of the algorithm. However, our main idea is to use representative sets to obtain $\widehat{\mathcal{S}}_t[Z]$ of small size. That is, given the set $\widehat{\mathcal{S}}_t[Z]$ that satisfies the correctness invariant, we use Theorem 3 to obtain a subset $\widehat{\mathcal{S}}'_t[Z]$ of $\widehat{\mathcal{S}}_t[Z]$ that also satisfies the correctness invariant and has size upper bounded by $2^{|Z|}$. Thus, we maintain the following size invariant.

> **Size Invariant:** After node t of $\mathbb{T}$ is processed by the algorithm, for every $Z \subseteq X_t$ we have that $|\widehat{\mathcal{S}}_t[Z]| \leq 2^{|Z|}$.

The new ingredient of the dynamic programming algorithm for STEINER TREE is the use of Theorem 3 to compute $\widehat{\mathcal{S}}_t[Z]$ maintaining the size invariant. The next lemma shows how to implement it.

Lemma 5.6 (Shrinking Lemma). *Let t be a node of $\mathbb{T}$, and let $Z \subseteq X_t$ be a set of size k. Furthermore, let $\widehat{\mathcal{S}}_t[Z]$ be a family of edge subsets of H_t satisfying the correctness invariant. If $|\widehat{\mathcal{S}}_t[Z]| = \ell$, then in time $\mathcal{O}\left(2^{k(\omega-1)}k^{\mathcal{O}(1)}\ell \cdot n\right)$ we can compute $\widehat{\mathcal{S}}'_t[Z] \subseteq \widehat{\mathcal{S}}_t[Z]$ satisfying correctness and size invariants.*

Proof. We start by associating a matroid with node t and the set $Z \subseteq X_t$ as follows. We consider a graphic matroid $M = (E, \mathcal{I})$ on $K^t[Z]$. Here, the element set E of the matroid is the edge set $E(K^t[Z])$ and the family of independent sets $\mathcal{I}$ consists of spanning forests of $K^t[Z]$.

Let $\widehat{\mathcal{S}}_t[Z] = \{E^t_1, \ldots, E^t_\ell\}$ and let $\mathcal{N} = \{F(E^t_1), \ldots, F(E^t_\ell)\}$ be the set of forests in $K^t[Z]$ corresponding to the edge subsets in $\widehat{\mathcal{S}}_t[Z]$. For $i \in \{1, \ldots, k-1\}$, let $\mathcal{N}_i$ be the family of forests of $\mathcal{N}$ with i edges. For each of family $\mathcal{N}_i$ we apply Theorem 3 and compute its min $(k-1-i)$-representative. That is,
$$\widehat{\mathcal{N}_i} \subseteq^{k-1-i}_{minrep} \mathcal{N}_i.$$

Let $\widehat{\mathcal{S}}'_t[Z] \subseteq \widehat{\mathcal{S}}_t[Z]$ be such that for every $E^t_j \in \widehat{\mathcal{S}}'_t[Z]$ we have that $F(E^t_j) \in \cup^{k-1}_{i=1}\widehat{\mathcal{N}_i}$. By Theorem 3, $|\widehat{\mathcal{S}}'_t[Z]| \leq \sum^{k-1}_{i=1} \binom{k}{i} \leq 2^k$. Now we show that the $\widehat{\mathcal{S}}'_t[Z]$ maintains the correctness invariant.

Let $L \in \mathscr{S}$ and let $L_t = E(H_t) \cap L$, $L_R = L \setminus L_t$ and $Z = \partial^t(L)$. Then there exists $E^t_j \in \widehat{\mathcal{S}}_t[Z]$ such that $w(E^t_j) \leq w(L_t)$, $\hat{L} = E^t_j \cup L_R$ is an optimal solution and $\partial^t(\hat{L}) = Z$. Consider the forest $F(E^t_j)$. Suppose its size is i, then $F(E^t_j) \in \mathcal{N}_i$. Now let $F(L_R)$ be the forest corresponding to L_R with respect to the bag X_t. Since $\hat{L}$ is a solution, we have that $F(E^t_j) \cup F(L_R)$ is a spanning tree in $K^t[Z]$. Since $\widehat{\mathcal{N}_i} \subseteq^{k-1-i}_{minrep} \mathcal{N}_i$, we have that there exists a forest $F(E^t_h) \in \widehat{\mathcal{N}_i}$ such that $w(F(E^t_h)) \leq w(F(E^t_i))$ and $F(E^t_h) \cup F(L_R)$ is a spanning tree in $K^t[Z]$. Thus, we know that $E^t_h \cup L_R$ is an optimum solution and $E^t_h \in \widehat{\mathcal{S}}'_t[Z]$. This proves that $\widehat{\mathcal{S}}'_t[Z]$ maintains the invariant.

The running time to compute $\widehat{\mathcal{S}}_t[Z]$ is dominated by:

$$\mathcal{O}\left(\sum_{i=1}^{k-1}\binom{k-1}{i}^{\omega-1}k^{\mathcal{O}(1)}\ell\right) = \mathcal{O}\left(2^{k(\omega-1)}k^{\mathcal{O}(1)}\ell\right).$$

For a given edge set we also need to compute the forest and that can take $\mathcal{O}(n)$ time. $\qquad\square$

In our algorithm the size of $\widehat{\mathcal{S}}_t[Z]$ can grow larger than $2^{|Z|}$ in intermediate steps but it will be at most $4^{|Z|}$ and thus we can use Shrinking Lemma (Lemma 5.6) to reduce its size efficiently.

We now return to the dynamic programming algorithm over the tree-decomposition $(\mathbb{T}, \mathcal{X})$ of G and prove that it maintains the correctness invariant. We assume that $(\mathbb{T}, \mathcal{X})$ is a nice tree-decomposition of G. By $\widehat{\mathcal{S}}_t$ we denote $\cup_{Z\subseteq X_t}\widehat{\mathcal{S}}_t[Z]$ (also called a *representative family of partial solutions*). We show how $\widehat{\mathcal{S}}_t$ is obtained by doing dynamic programming from base node to the root node.

Base node t. Here the graph H_t is empty and thus we take $\widehat{\mathcal{S}}_t = \emptyset$.

Introduce node t with child t'. Here, we know that $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$. Let v be the vertex in $X_t \setminus X_{t'}$. Furthermore observe that $E(H_t) = E(H_{t'})$ and v is degree zero vertex in H_t. Thus the graph H_t only differs from $H_{t'}$ at a isolated vertex v. Since we have not added any edge to the new graph, the family of solutions, which contains edge-subsets, does not change. Thus, we take $\widehat{\mathcal{S}}_t = \widehat{\mathcal{S}}_{t'}$. Formally, we take $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t'}[Z \setminus \{v\}]$. Since, H_t and $H_{t'}$ have same set of edges the invariant is vacuously maintained.

Forget node t with child t'. Here we know $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$. Let v be the vertex in $X_{t'} \setminus X_t$. Let $\mathcal{E}_v[Z]$ denote the set of edges between v and the vertices in $Z \subseteq X_t$. Observe that $E(H_t) = E(H_{t'}) \cup \mathcal{E}_v[X_t]$. Before we define things formally, observe that in this step the graphs H_t and $H_{t'}$ differ by at most **tw** edges - the edges with one endpoint in v and the other in X_t. We go through every possible way an optimal solution can intersect with these newly added edges. The idea is that for every edge subset in our family of partial solutions we make several new partial solutions, one each for every subset of newly added edges. More formally the new set of partial solutions is defined as follows. Let

$$\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t'}[Z] \bigcup_{X \subseteq \mathcal{E}_v[Z]} \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \circ X.$$

Recall that for two families $\mathcal{A}$ and $\mathcal{B}$, we defined $\mathcal{A} \circ \mathcal{B} = \{A \cup B \ : \ A \in \mathcal{A} \wedge B \in \mathcal{B}\}$.

Now we show that $\widehat{\mathcal{S}}_t$ maintains the invariant of the algorithm. Let $L \in \mathscr{S}$.

1. Let $L_t = E(H_t) \cap L$ and $L_R = L \setminus L_t$. Furthermore, edges of L_t can be partitioned into $L_{t'} = E(H_{t'}) \cap L$ and $L_v = L_t \setminus L_{t'}$. That is, $L_t = L_{t'} \uplus L_v$.

2. Let $Z = \partial^t(L)$ and $Z' = \partial^{t'}(L)$.

By the property of $\widehat{\mathcal{S}}_{t'}$, there exists a $\hat{L}_{t'} \in \widehat{\mathcal{S}}_{t'}[Z']$ such that

$$\begin{aligned} L \in \mathscr{S} &\iff L_{t'} \uplus L_v \uplus L_R \in \mathscr{S} \\ &\iff \hat{L}_{t'} \uplus L_v \uplus L_R \in \mathscr{S} \end{aligned} \tag{5}$$

and $\partial^{t'}(L) = \partial^{t'}(\hat{L}_{t'} \uplus L_v \uplus L_R) = Z'$.

We put $\hat{L}_t = \hat{L}_{t'} \cup L_v$ and $\hat{L} = \hat{L}_t \cup L_R$. We know show that $\hat{L}_t \in \widehat{\mathcal{S}}_t[Z]$. Towards this just note that since $Z' = Z$ or $Z' = Z \cup \{v\}$, we have that $\widehat{\mathcal{S}}_t[Z]$ contains $\widehat{\mathcal{S}}_{t'}[Z'] \circ L_v$. By (5), $\hat{L} \in \mathscr{S}$. Finally, we need to show that $\partial^t(\hat{L}) = Z$. Towards this just note that $\partial^t(\hat{L}) = Z' \setminus \{v\} = Z$. This concludes the proof for the fact that $\widehat{\mathcal{S}}_t$ maintains the correctness invariant.

Join node t with two children t_1 and t_2.   Here, we know that $X_t = X_{t_1} = X_{t_2}$. Also we know that the edges of H_t is obtained by the union of edges of H_{t_1} and H_{t_2} which are disjoint. Of course they are separated by the vertices in X_t. A natural way to obtain a family of partial solutions for H_t is that we take the union of edges subsets of the families stored at nodes t_1 and t_2. This is exactly what we do. Let

$$\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t_1}[Z] \circ \widehat{\mathcal{S}}_{t_2}[Z].$$

Now we show that $\widehat{\mathcal{S}}_t$ maintains the invariant. Let $L \in \mathscr{S}$.

1. Let $L_t = E(H_t) \cap L$ and $L_R = L \setminus L_t$. Furthermore edges of L_t can be partitioned into those belonging to H_{t_1} and those belonging to H_{t_2}. Let $L_{t_1} = E(H_{t_1}) \cap L$ and $L_{t_2} = E(H_{t_2}) \cap L$. Observe that since $E(H_{t_1}) \cap E(H_{t_2}) = \emptyset$, we have that $L_{t_1} \cap L_{t_2} = \emptyset$. Also observe that $L_t = L_{t_1} \uplus L_{t_2}$.

2. Let $Z = \partial^t(L)$. Since $X_t = X_{t_1} = X_{t_2}$ this implies that $Z = \partial^t(L) = \partial^{t_1}(L) = \partial^{t_2}(L)$.

Now observe that

$$
\begin{aligned}
L \in \mathscr{S} &\iff L_{t_1} \uplus L_{t_2} \uplus L_R \in \mathscr{S} \\
&\iff \hat{L}_{t_1} \uplus L_{t_2} \uplus L_R \in \mathscr{S} \quad \text{(by the property of } \widehat{\mathcal{S}}_{t_1} \text{ we have that } \hat{L}_{t_1} \in \widehat{\mathcal{S}}_{t_1}[Z]) \\
&\iff \hat{L}_{t_1} \uplus \hat{L}_{t_2} \uplus L_R \in \mathscr{S} \quad \text{(by the property of } \widehat{\mathcal{S}}_{t_2} \text{ we have that } \hat{L}_{t_2} \in \widehat{\mathcal{S}}_{t_2}[Z])
\end{aligned}
$$

We put $\hat{L}_t = \hat{L}_{t_1} \cup \hat{L}_{t_2}$. By the definition of $\widehat{\mathcal{S}}_t[Z]$, we have that $\hat{L}_{t_1} \cup \hat{L}_{t_2} \in \widehat{\mathcal{S}}[Z]$. The above inequalities also show that $\hat{L} = \hat{L}_t \cup L_R \in \mathscr{S}$. It remains to show that $\partial^t(\hat{L}) = Z$. Since $\partial^{t_1}(L) = Z$, we have that $\partial^{t_1}(\hat{L}_{t_1} \uplus L_{t_2} \uplus L_R) = Z$. Now since $X_{t_1} = X_{t_2}$ we have that $\partial^{t_2}(\hat{L}_{t_1} \uplus L_{t_2} \uplus L_R) = Z$ and thus $\partial^{t_2}(\hat{L}_{t_1} \uplus \hat{L}_{t_2} \uplus L_R) = Z$. Finally, because $X_{t_2} = X_t$, we conclude that $\partial^t(\hat{L}_{t_1} \uplus \hat{L}_{t_2} \uplus L_R) = \partial^t(\hat{L}) = Z$. This concludes the proof of correctness invariant.

Root node r.   Here, $X_r = \emptyset$. We go through all the solution in $\widehat{\mathcal{S}}_r[\emptyset]$ and output the one with the minimum weight. This concludes the description of the dynamic programming algorithm.

Computation of $\widehat{\mathcal{S}}_t$.   Now we show how to implement the algorithm described above in the desired running time by making use of Lemma 5.6. For our discussion let us fix a node t and $Z \subseteq X_t$ of size k. While doing dynamic programming algorithm from the base nodes to the root node we always maintain the size invariant. That is, $\widehat{\mathcal{S}}_t[Z]| \le 2^k$.

Base node t.   Trivially, in this case we have $|\widehat{\mathcal{S}}_t[Z]| \le 2^k$.

Introduce node t with child t'.   Here, we have that $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t'}[Z \setminus \{v\}]$ and thus $|\widehat{\mathcal{S}}_t[Z]| = |\widehat{\mathcal{S}}_{t'}[Z \setminus \{v\}]| \le 2^{k-1} \le 2^k$.

Forget node t with child t'.   In this case,

$$\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t'}[Z] \bigcup_{X \subseteq \mathcal{E}_v[Z]} \widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \circ X.$$

Observe that,

$$|\widehat{\mathcal{S}}_t[Z]| = |\widehat{\mathcal{S}}_{t'}[Z]| + \sum_{X \subseteq \mathcal{E}_v[Z]} |\widehat{\mathcal{S}}_{t'}[Z \cup \{v\}] \circ X| \le 2^k + \sum_{i=1}^{k} \binom{k}{i} 2^{k+1} = \mathcal{O}(4^k).$$

It can happen in this case that the size of $\widehat{\mathcal{S}}_t[Z]$ is larger than 2^k and thus we need to reduce the size of family. We apply Lemma 5.6 and obtain $\widehat{\mathcal{S}}'_t[Z]$ that maintains the correctness and size invariants. We update $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}'_t[Z]$.

The running time to compute $\widehat{\mathcal{S}}_t$ (that is, across all subsets of X_t) is

$$\mathcal{O}\left(\sum_{i=1}^{\mathbf{tw}+1}\binom{\mathbf{tw}+1}{i}2^{i(\omega-1)}4^i \cdot \mathbf{tw}^{\mathcal{O}(1)}n\right) = \mathcal{O}\left((1+2^{\omega+1})^{\mathbf{tw}} \cdot \mathbf{tw}^{\mathcal{O}(1)}n\right).$$

Join node t with two children t_1 and t_2.   Here we defined

$$\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}_{t_1}[Z] \circ \widehat{\mathcal{S}}_{t_2}[Z].$$

The size of $\widehat{\mathcal{S}}_t[Z]$ is $2^k \cdot 2^k = 4^k$. Now, we apply Lemma 5.6 and obtain $\widehat{\mathcal{S}}'_t[Z]$ that maintains the correctness invariant and has size at most 2^k. We put $\widehat{\mathcal{S}}_t[Z] = \widehat{\mathcal{S}}'_t[Z]$.

The running time to compute $\widehat{\mathcal{S}}_t$ is

$$\mathcal{O}\left(\sum_{i=1}^{\mathbf{tw}+1}\binom{\mathbf{tw}+1}{i}4^i 2^{i\omega-1} \cdot \mathbf{tw}^{\mathcal{O}(1)}n\right) = \mathcal{O}\left((1+2^{\omega+1})^{\mathbf{tw}} \cdot \mathbf{tw}^{\mathcal{O}(1)}n\right).$$

Thus the whole algorithm takes $\mathcal{O}\left((1+2^{\omega+1})^{\mathbf{tw}} \cdot \mathbf{tw}^{\mathcal{O}(1)} \cdot n^2\right)$ as the number of nodes in a nice tree-decomposition is upper bounded by $\mathcal{O}(n)$. However, observe that we do not need to compute the forests and the associated weight at every step of the algorithm. The size of the forest is at most $\mathbf{tw}+1$ and we can maintain these forests across the bags during dynamic programming in time $\mathbf{tw}^{\mathcal{O}(1)}$. This will lead to an algorithm with the claimed running time. The last remark we would like to make is that one can do better at **forget node** by forgetting a single edge at a time. However, we did not try to optimize this, as the running time to compute the family of partial solutions at **join node** is the most expensive operation. This completes the proof. $\square$

The approach of Theorem 9 can be used to obtain single-exponential algorithms parameterized by the treewidth of an input graph for many other connectivity problems such as HAMILTONIAN CYCLE, FEEDBACK VERTEX SET, and CONNECTED DOMINATED SET. For all these problems, checking whether two partial solutions can be glued together to form a global solution can be checked by testing independence in a specific graphic matroid. We believe that there exist interesting problems where this check corresponds to testing independence in a different class of linear matroids.

5.4   Path, Trees and Subgraph Isomorprhism

In this section we outline algorithms for k-PATH, k-TREE and k-SUBGRAPH ISOMORPHISM using representative sets. All results in this section are based on computing representative families with respect to uniform matroids.

5.4.1 k-PATH

The problem we study in this section is as follows.

---

k-PATH **Parameter:** k
Input: An undirected n-vertex and m-edge graph G and a positive integers k.
Question: Does there exist a simple path of length k in G?

We start by modifying the graph slightly. We add a new vertex, say s not present in $V(G)$, to G by making it adjacent to every vertex in $V(G)$. Let the modified graph be called G'. It is clear that G has a path of length k if and only if G' has a path of length $k+1$ starting from s. For ease of presentation we rename G' to G and the objective is to find a path of length $k+1$ starting from s. Let $M = (E, \mathcal{I})$ be an uniform matroid $U_{n,k+2}$ where $E = V(G)$ and $\mathcal{I} = \{S \subseteq V(G) \mid |S| \leq k+2\}$. In this section whenever we speak about independent sets we mean independence with respect to the uniform matroid $U_{n,k+2}$ defined above. For a given pair of vertices $s, v \in V(G)$, recall that we defined

$$\mathcal{P}^i_{sv} \;=\; \Big\{ X \;\Big|\; X \subseteq V(G), \; v, s \in X, \; |X| = i \text{ and there is a path from } s \text{ to } v \text{ of length } i$$
$$\text{in } G \text{ with all the vertices belonging to } X. \Big\}$$

Now using Lemma 5.2 we compute $\widehat{\mathcal{P}}^{k+2}_{sv} \subseteq^{k+2}_{rep} \mathcal{P}^{k+2}_{sv}$ for all $v \in V(G) \setminus \{s\}$ in time

$$\mathcal{O}\left(2^{o(k)} m \log^2 n \max_{i \in [k+2]} \left\{ \binom{k+2}{i-1} \left(\frac{k+2}{k+2-i} \right)^{k+2-i} \right\} \right).$$

The maximum is achieved at $i = \alpha k$ where $\alpha = 1 + \frac{1 - \sqrt{1+4e}}{2e}$. Thus, the running time of the algorithm is $\mathcal{O}(2.8505^k \cdot 2^{o(k)} m \log^2 n) = \mathcal{O}(2.851^k \cdot m \log^2 n)$.

Furthermore, in the same time every set in $\widehat{\mathcal{P}}^p_{uv}$ can be ordered in a way that it corresponds to an undirected path in G. A graph G has a path of length $k+1$ starting from s if and only if for some $v \in V(G) \setminus \{s\}$, we have that $\widehat{\mathcal{P}}^{k+2}_{sv} \neq \emptyset$. Thus the running time of this algorithm is upper bounded by $\mathcal{O}(2.851^k \cdot m \log^2 n)$. Let us remark that almost the same arguments show that the version of the problem on directed graphs is solvable within the same running time. However on undirected graphs we can speed up the algorithm slightly by using the following standard trick. We need the following result.

Proposition 5.3 ([6])**.** *There exists an algorithm, that given a graph G and an integer k, in time $\mathcal{O}(k^2 n)$ either finds a simple path of length $\geq k$ or computes a DFS (depth first search) tree rooted at some vertex of G of depth at most k.*

We first apply Proposition 5.3 and in time $\mathcal{O}(k^2 n)$ either find a simple path of length $\geq k$ in G or compute a DFS tree of G of depth at most k. In the former case we simply output the same path. In the later case since all the root to leaf paths are upper bounded by k and there are no cross edges in a DFS tree, we have that the number of edges in G is upper bounded by $\mathcal{O}(k^2 n)$. Now on this G we apply the representative set based algorithm described above. This results in the following theorem.

Theorem 10. k-Path *can be solved in time* $\mathcal{O}(2.851^k \cdot n \log^2 n)$.

5.4.2 k-Tree **and** k-Subgraph Isomorphism

In this section we consider the following problem.

---
k-Tree **Parameter:** k
Input: An undirected n-vertex, m-edge graph G and a tree T on k vertices.
Question: Does G contains a subgraph isomorphic to T?

We design an algorithm for k-Tree using the method of representative sets. The algorithm for k-Tree is more involved than for k-Path. The reason to that is due to the fact that paths poses perfectly balanced separators of size one while trees not. We select a leaf r of T and root

the tree at r. For vertices $x, y \in V(T)$ we say that $y \leq x$ if x lies on the path from y to r in T (if $x = r$ we also say that $y \leq x$). For a set C of vertices in T we will say that $x \preceq_C y$ if $x \leq y$ and there is no $z \in C$ such that $x \leq z$ and $z \leq y$. For a pair x, y of vertices such that $y \leq x$ in T we define

$$C^{xy} = \begin{cases} \emptyset & \text{if } xy \in E(T), \\ \text{The unique component } C \text{ of } T \setminus \{x, y\} \text{ such that } N(C) = \{x, y\} & \text{otherwise.} \end{cases}$$

We also define $T^{uv} = T[C^{uv} \cup \{u, v\}]$. We start by making a few simple observations about sets of vertices in trees.

Lemma 5.7. *For any tree T, a pair $\{x, y\}$ of vertices in $V(T)$ and integer $c \geq 1$ there exists a set W of vertices such that $\{x, y\} \subseteq W$, $|W| = \mathcal{O}(c)$ and every connected component U of $T \setminus W$ satisfies $|U| \leq \frac{|V(T)|}{c}$ and $|N(U)| \leq 2$.*

Proof. We first find a set W^1 of size at most c such that every connected component U of $T \setminus W$ satisfies $|U| \leq \frac{|V(T)|}{c}$. Start with $W^1 = \emptyset$ and select a lowermost vertex $u \in V(T)$ such that the subtree rooted at u has at least $\frac{|V(T)|}{c}$ vertices. Add u to W^1 and remove the subtree rooted at u from T. The process must stop after c iterations since each iteration removes $\frac{|V(T)|}{c}$ vertices of T. Each component U of $T \setminus W^1$ satisfies $|U| \leq \frac{|V(T)|}{c}$ because (a) whenever a vertex u is added to W^1, all components below u have size strictly less than $\frac{|V(T)|}{c}$ and (b) when the process ends the subtree rooted at r has size at most $|U| \leq \frac{|V(T)|}{c}$. Now, insert x and y into W^1 as well.

We build W from W^1 by taking the *least common ancestor closure* of W_1; start with $W = W_1$ and as long as there exist two vertices u and v in W such that their least common ancestor w is not in W, add w to W. Standard counting arguments on trees imply that this process will never increase the size of W by more than a factor 2, hence $|W| \leq 2|W_1| = O(c)$.

We claim that every connected component U of $T \setminus W$ satisfies $N(U) \leq 2$. Suppose not and let u be the vertex of u closest to the root. Since $N(U) > 2$ at least two vertices v and w in $N(U)$ are descendants of u. Since U is connected v and w can't be descendants of each other, but then the least common ancestor of v and w is in U, contradicting the construction of W. $\qquad\square$

Observation 5.1. *For any tree T, set $W \subseteq V(T)$ and component U of $T \setminus W$ such that $|N(U)| = 1$, U contains a leaf of T.*

Proof. $T[U \cup N(U)]$ is a tree on at least two vertices and hence it has at least two leaves. At most one of these leaves is in $N(U)$, the other one is also a leaf of T. $\qquad\square$

Lemma 5.8. *Let $W \subseteq V(T)$ be a set of vertices such that for every pair of vertices in W their least common ancestor is also in W. Let X be a set containing one leaf of T from each connected component U of $T \setminus W$ such that $|N(U)| = 1$. Then, for every connected component U such that $|N(U)| = 1$ there exist $x \in W$, $y \in X$ such that $U = C^{xy} \cup \{y\}$. For every other connected component U there exist $x, y \in W$ such that $U = C^{xy}$.*

Proof. It follows from the argument at the end of the proof of Lemma 5.7 that every component U of $T \setminus W$ satisfies $|N(U)| \leq 2$. If $|N(U)| = 2$, let $N(U) = \{x, y\}$. We have that $x \leq y$ or $y \leq x$ since least common ancestor of x and y can not be in U and would therefore be in $N(U)$, contradicting $|N(U)| = 2$. Without loss of generality $y \leq x$. But then $U = C^{xy}$. If $N(U) = 1$, let $N(U) = \{x\}$. By Observation 5.1 U contains a leaf y of T. Then $U = C^{xy} \cup \{y\}$. $\qquad\square$

Given two graphs F and H, a graph *homomorphism* from F to H is a map f from $V(F)$ to $V(H)$, that is $f : V(F) \to V(H)$, such that if $uv \in E(F)$, then $f(u)f(v) \in E(H)$. Furthermore, when the map f is injective and $|V(H)| = |V(F)|$, f is called a *subgraph isomorphism*. For every $x, y \in V(T)$ such that $y \le x$, and every u,v in $V(G)$ we define

$$\mathcal{F}_{uv}^{xy} \;=\; \left\{ F \in \binom{V(G) \setminus \{u, v\}}{|C^{uv}|} \;:\; \exists \text{ subgraph isomorphism } f \right.$$
$$\left. \text{from } T^{xy} \text{ to } G[F \cup \{u, v\}] \text{ such that } f(x) = u \text{ and } f(y) = v \right\}$$

For every $x, y \in V(T)$ such that $y \le x$, and every u in $V(G)$ we define

$$\mathcal{F}_{u*}^{xy} = \bigcup_{v \in V(G) \setminus \{u\}} \mathcal{F}_{uv}^{xy} \oplus \{v\} \tag{6}$$

In order to solve the problem it is sufficient to select an arbitrary leaf ℓ of t and determine whether there exists a $u \in V(G)$ such that the family $\mathcal{F}_{u*}^{r\ell}$ is non-empty. We show that the collections of families $\{\mathcal{F}_{uv}^{xy}\}$ and $\{\mathcal{F}_{u*}^{xy}\}$ satisfy a recurrence relation. We will then exploit this recurrence relation to get a fast algorithm for k-TREE.

Lemma 5.9. *For every $x, y \in V(T)$ such that $y \le x$, every $\widehat{W} = W \cup \{x, y\}$ where $W \subseteq C^{xy}$, such that for every pair of vertices in $\widehat{W}$ their least common ancestor is also in $\widehat{W}$, every $X \subseteq C^{uv} \setminus W$ such that X contains exactly one leaf of T in each connected component U of $T^{xy} \setminus \widehat{W}$ with $|N(U)| = 1$, the following recurrence holds.*

$$\mathcal{F}_{uv}^{xy} = \bigcup_{\substack{g:\widehat{W}\to V(G) \\ g(x)=u \wedge g(y)=v}} \left[\left(\overset{\bullet}{\prod_{\substack{x',y'\in\widehat{W} \\ y' \preceq_{\widehat{W}} x'}}} \mathcal{F}_{g(x')g(y')}^{x'y'} \bullet \overset{\bullet}{\prod_{\substack{x'\in\widehat{W},\, y'\in X \\ y' \preceq_{\widehat{W}} x'}}} \mathcal{F}_{g(x')*}^{x'y'} \right) \oplus g(W) \right] \tag{7}$$

Here the union goes over all $O(n^{|W|})$ injective maps g from $\widehat{W}$ to $V(G)$ such that $g(x) = u$ and $g(y) = v$, and by $g(W)$ we mean $\{g(c) \;:\; c \in W\}$.

Proof. For the $\subseteq$ direction of the equality consider any subgraph isomorphism f from T^{xy} to $V(G)$ such that $f(x) = u$ and $f(y) = v$. Let g be the restriction of f to W. The map f can be considered as a collection of subgraph isomorphisms with one isomorphism for each $x', y' \in \widehat{W}$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$ and $f(y') = g(y')$, and one isomorphism for each $x' \in \widehat{W}, y' \in X$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$. Taking the union of the ranges of each of the small subgraph isomorphisms clearly give the range of f. Here we used Lemma 5.8 to argue that for every connected component U of $T^{xy} \setminus \widehat{W}$ we have that $T[U \cup N(U)]$ is in fact on the form $T^{x'y'}$ for some x', y'.

For the reverse direction take any collection of subgraph isomorphisms with one isomorphism f for each $x', y' \in \widehat{W}$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$ and $f(y') = g(y')$, and one isomorphism for each $x' \in \widehat{W}, y' \in X$ such that $y' \preceq_{\widehat{W}} x$ from $T^{x'y'}$ to G such that $f(x') = g(x')$, such that the range of all of these subgraph isomorphisms are pairwise disjoint (except on vertices in $\widehat{W}$). Since all of these subgraph isomorphisms agree on the set W they can be glued together to a subgraph isomorphism from T^{xy} to G. $\qquad\square$

Our goal is to compute for every $x, y \in V(T)$ such that $y \le x$ and $u, v \in V(G)$ a family $\hat{\mathcal{F}}_{uv}^{xy}$ such that $\hat{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \mathcal{F}_{uv}^{xy}$ and for every $x, y \in V(T)$ such that $y \le x$ and $u \in V(G)$ a

family $\hat{\mathcal{F}}_{u*}^{xy}$ such that $\hat{\mathcal{F}}_{u*}^{xy} \subseteq_{rep}^{k-|C^{xy}|-1} \mathcal{F}_{uv}^{xy}$. We will also maintain the invariant that $|\hat{\mathcal{F}}_{uv}^{xy}| \leq \binom{k}{|C^{xy}|}2^{o(k)}\log n$ and $|\hat{\mathcal{F}}_{u*}^{xy}| \leq \binom{k}{|C^{xy}|+1}2^{o(k)}\log n$.

We first compute such families $\hat{\mathcal{F}}_{uv}^{xy}$ for all $x,y \in V(T)$ such that $y \leq x$ and $xy \in E(T)$. Observe that in this case we have

$$\mathcal{F}_{uv}^{xy} = \begin{cases} \{\emptyset\} & \text{if } uv \in E(G), \\ \emptyset & \text{if } uv \notin E(G). \end{cases}$$

For each $x,y \in V(T)$ such that $y \leq x$ and $xy \in E(T)$ and every $u,v \in V(G)$ we set $\hat{\mathcal{F}}_{uv}^{xy} = \mathcal{F}_{uv}^{xy}$. We can now for compute $\hat{\mathcal{F}}_{u*}^{xy}$ for every $x,y \in V(T)$ such that $y \leq x$ and $xy \in E(T)$ and every $u \in V(G)$ by applying Equation 6. Clearly the computed families are within the required size bounds.

We now show how to compute a family $\hat{\mathcal{F}}_{uv}^{xy}$ of size $\binom{k}{|C^{xy}|}2^{o(k)}\log n$ for every $x,y \in V(T)$ such that $y \leq x$ and $u,v \in V(G)$ and $|C^{xy}| = t$, assuming that the families $\hat{\mathcal{F}}_{uv}^{xy}$ and $\hat{\mathcal{F}}_{u*}^{xy}$ have been computed for every $x,y \in V(T)$ such that $y \leq x$ and $u,v \in V(G)$ and $|C^{xy}| < t$. We also assume that for each family $\hat{\mathcal{F}}_{uv}^{xy}$ that has been computed $|\hat{\mathcal{F}}_{uv}^{xy}| \leq \binom{k}{|C^{xy}|}2^{o(k)}\log n$. Similarly we assume that for each family $\hat{\mathcal{F}}_{u*}^{xy}$ that has been computed $|\hat{\mathcal{F}}_{u*}^{xy}| \leq \binom{k}{|C^{xy}|+1}2^{o(k)}\log n$.

We fix a constant c whose value will be decided later. First apply Lemma 5.7 on T^{xy}, vertex pair $\{x,y\}$ and constant c and obtain a set $\widehat{W}$ such that $\{x,y\} \subseteq \widehat{W}$ and every connected component U of $T \setminus \widehat{W}$ satisfies $|U| \leq \frac{|V(T)|}{c}$ and $|N(U)| \leq 2$. Select a set $X \subseteq V(T^{x,y}) \setminus \widehat{W}$ such that each connected component U of $T \setminus \widehat{W}$ with $|N(U)| = 1$ contains exactly one leaf which is in X. Now, set $W = \widehat{W} \setminus \{x,y\}$ and consider Equation 7 for $\hat{\mathcal{F}}_{uv}^{xy}$ for this choice of x,y,W and X. Define

$$\tilde{\mathcal{F}}_{uv}^{xy} = \bigcup_{\substack{g:\widehat{W} \to V(G) \\ g(x)=u \wedge g(y)=v}} \left[\left(\overset{\bullet}{\prod_{\substack{x',y' \in \widehat{W} \\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}_{g(x')g(y')}^{x'y'} \bullet \overset{\bullet}{\prod_{\substack{x' \in \widehat{W}, \ y' \in X \\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}_{g(x')*}^{x'y'} \right) \oplus g(W) \right] \tag{8}$$

Lemma 5.9 together with Lemmata 3.2 and 3.3 directly imply that $\tilde{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \mathcal{F}_{uv}^{xy}$. Furthermore, each family on the right hand side of Equation 8 has already been computed, since $C^{x'y'} \subset C^{xy}$ and so $|C^{x'y'}| < t$. For a fixed injective map $g : W \to V(G)$ we define

$$\tilde{\mathcal{F}}_{g}^{xy} = \left(\overset{\bullet}{\prod_{\substack{x',y' \in \widehat{W} \\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}_{g(x')g(y')}^{x'y'} \bullet \overset{\bullet}{\prod_{\substack{x' \in \widehat{W}, \ y' \in X \\ y' \preceq_{\widehat{W}} x'}}} \hat{\mathcal{F}}_{g(x')*}^{x'y'} \right) \oplus g(W) \tag{9}$$

It follows directly from the definition of $\tilde{\mathcal{F}}_{uv}^{xy}$ and $\tilde{\mathcal{F}}_g^{xy}$ that

$$\tilde{\mathcal{F}}_{uv}^{xy} = \bigcup_{\substack{g:\widehat{W} \to V(G) \\ g(x)=u \wedge g(y)=v}} \tilde{\mathcal{F}}_g^{xy}.$$

Our goal is to compute a family $\hat{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \tilde{\mathcal{F}}_{uv}^{xy}$ such that $|\hat{\mathcal{F}}_{uv}^{xy}| \leq \binom{k}{|C^{xy}|}2^{o(k)}\log n$. Lemma 3.1 then implies that $\hat{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \mathcal{F}_{uv}^{xy}$. To that end, we define the function reduce. Given a family $\mathcal{F}$ of sets of size p, the function reduce will run the algorithm of Theorem 2 on $\mathcal{F}$ and produce a family of size $\binom{k}{p} \cdot 2^{o(k)} \cdot \log n$ that $k-p$ represents $\mathcal{F}$.

We will compute for each $g : \widehat{W} \to V(G)$ such that $g(x) = u$ and $g(y) = v$ a family $\hat{\mathcal{F}}_g^{xy}$ of size at most $\binom{k}{|C^{xy}|} 2^{o(k)} \log n$ such that $\hat{\mathcal{F}}_g^{xy} \subseteq_{rep}^{k-|C^{xy}|} \tilde{\mathcal{F}}_g^{xy}$. We will then set

$$\hat{\mathcal{F}}_{uv}^{xy} = \mathsf{reduce} \left(\bigcup_{\substack{g:\widehat{W}\to V(G) \\ g(x)=u \wedge g(y)=v}} \hat{\mathcal{F}}_g^{xy} \right). \tag{10}$$

To compute $\hat{\mathcal{F}}_g^{xy}$, inspect Equation 9. Equation 9 shows that $\tilde{\mathcal{F}}_g^{xy}$ basically is a long chain of $\bullet$ operations, specifically

$$\tilde{\mathcal{F}}_g^{xy} = \left(\hat{F}_1 \bullet \hat{F}_2 \bullet \hat{F}_3 \ldots \bullet \hat{F}_\ell \right) \oplus g(W) \tag{11}$$

We define (and compute) $\hat{\mathcal{F}}_g^{xy}$ as follows

$$\hat{\mathcal{F}}_g^{xy} = \mathsf{reduce} \left(\mathsf{reduce} \left(\ldots \mathsf{reduce} \left(\mathsf{reduce} \left(\hat{F}_1 \bullet \hat{F}_2 \right) \bullet \hat{F}_3 \right) \bullet \ldots \right) \bullet \hat{F}_\ell \right) \oplus g(W) \tag{12}$$

$\hat{\mathcal{F}}_g^{xy} \subseteq_{rep}^{k-|C^{xy}|} \tilde{\mathcal{F}}_g^{xy}$ and thus also $\hat{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \tilde{\mathcal{F}}_{uv}^{xy} \subseteq_{rep}^{k-|C^{xy}|} \mathcal{F}_{uv}^{xy}$ follows from Lemma 3.3 and Theorem 2. Since the last operation we do in the construction of $\hat{\mathcal{F}}_{uv}^{xy}$ is a call to reduce, $|\hat{\mathcal{F}}_{uv}^{xy}| \leq \binom{k}{|C^{xy}|} 2^{o(k)} \log n$ follows from Theorem 2. To conclude the computation we set

$$\tilde{\mathcal{F}}_{u*}^{xy} = \mathsf{reduce} \left(\bigcup_{v \in V(G) \setminus \{u\}} \hat{\mathcal{F}}_{uv}^{xy} \oplus \{v\} \right) \tag{13}$$

Lemma 3.3 and Theorem 2 imply that $\tilde{\mathcal{F}}_{u*}^{xy} \subseteq_{rep}^{k-|C^{xy}|-1} \mathcal{F}_{u*}^{xy}$ and that $|\hat{\mathcal{F}}_{u*}^{xy}| \leq \binom{k}{|C^{xy}|+1} 2^{o(k)} \log n$.

The algorithm computes the families $\hat{\mathcal{F}}_{u*}^{xy}$ and $\hat{\mathcal{F}}_{uv}^{xy}$ for every $x, y \in V(T)$ such that $y \leq x$. It then selects an arbitrary leaf ℓ of T and checks whether there exists a $u \in V(G)$ such that the family $\hat{\mathcal{F}}_{u*}^{r\ell}$ is non-empty. Since $\hat{\mathcal{F}}_{u*}^{r\ell} \subseteq_{rep}^0 \mathcal{F}_{u*}^{r\ell}$ there is a non-empty $\mathcal{F}_{u*}^{r\ell}$ if and only if there is a non empty $\hat{\mathcal{F}}_{u*}^{r\ell}$. Thus the algorithm can answer that there is a subgraph isomorphism from T to G if some $\hat{\mathcal{F}}_{u*}^{r\ell}$ is non-empty, and that no such subgraph isomorphism exists otherwise.

It remains to bound the running time of the algorithm. Up to polynomial factors, the running time of the algorithm is dominated by the computation of $\hat{\mathcal{F}}_{uv}^{xy}$. This computation consists of $n^{\mathcal{O}(|\widehat{W}|)}$ independent computations of the families $\hat{\mathcal{F}}_g^{xy}$. Each computation of the family $\hat{\mathcal{F}}_g^{xy}$ consists of at most k repeated applications of the operation

$$\hat{\mathcal{F}}^{i+1} = \mathsf{reduce}(\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}).$$

Here $\mathcal{F}^i$ is a family of sets of size p, and so $|\mathcal{F}^i| \leq \binom{k}{p} 2^{o(k)} \log n$. On the other hand $\hat{\mathcal{F}}_{i+1}$ is a family of sets of size $p' \leq \frac{k}{c}$ since we used Lemma 5.7 to construct $\widehat{W}$. Thus $|\hat{\mathcal{F}}_{i+1}| \leq \binom{k}{k/c} 2^{o(k)} \log n \leq 2^{\varepsilon k} \log n$. Thus $|\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1}| \leq \binom{k}{p} 2^{\varepsilon k + o(k)} \log^2 n$. Hence, when we apply Theorem 2 to compute $\mathsf{reduce}(\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1})$, this takes time

$$\binom{k}{p} \left(\frac{k}{k-p-p'} \right)^{k-p-p'} 2^{\varepsilon k + o(k)} n^{\mathcal{O}(1)}$$

$$\leq \binom{k}{p} \left(\frac{k}{k-p} \right)^{k-p} \cdot \left(\frac{k-p}{k-p-p'} \right)^{k-p-p'} 2^{\varepsilon k + o(k)} n^{\mathcal{O}(1)}$$

For $p \geq \frac{9k}{10}$ this is upper bounded by 2^k, regardless the choice of p'. For $p \leq \frac{9k}{10}$ we choose c such that

$$p' \leq \frac{k}{c} \leq \frac{\varepsilon k}{10} \leq (k - p)\frac{\varepsilon}{1 + \varepsilon},$$

and then the running time of one $\mathsf{reduce}(\hat{\mathcal{F}}^i \bullet \hat{\mathcal{F}}_{i+1})$ step is upper bounded by

$$\binom{k}{p}\left(\frac{k}{k-p}\right)^{k-p} \cdot (1+\varepsilon)^k \cdot 2^{\varepsilon k + o(k)} n^{\mathcal{O}(1)} \leq \binom{k}{p}\left(\frac{k}{k-p}\right)^{k-p} \cdot 2^{3\varepsilon k + o(k)} n^{\mathcal{O}(1)}$$

As the running time is dominated by $n^{\mathcal{O}(|\widehat{W}|)} = n^{\mathcal{O}(c)}$ such reduction steps and $c = \mathcal{O}(\frac{1}{\varepsilon})$ the total running time of the algorithm is upper bounded by

$$\max_{p \leq k}\left(\binom{k}{p}\left(\frac{k}{k-p}\right)^{k-p}\right) \cdot 2^{3\varepsilon k} n^{\mathcal{O}(\frac{1}{\varepsilon})}.$$

It is possible to show that for $\alpha = 1 + \frac{1-\sqrt{1+4e}}{2e}$,

$$\max_{p \leq k}\left(\binom{k}{p}\left(\frac{k}{k-p}\right)^{k-p}\right) \leq \binom{k}{\alpha k}\left(\frac{1}{1-\alpha}\right)^{k(1-\alpha)} \leq 2.851^k.$$

This yields the following theorem.

Theorem 11. *For every $\varepsilon > 0$, k-TREE is solvable in time $\binom{k}{\alpha k}\left(\frac{1}{1-\alpha}\right)^{k(1-\alpha)} \cdot 2^{3\varepsilon k} n^{\mathcal{O}(\frac{1}{\varepsilon})}$, where $\alpha = 1 + \frac{1-\sqrt{1+4e}}{2e}$. For sufficiently small ε this is upper bounded by $2.851^k n^{\mathcal{O}(1)}$.*

The algorithm for k-TREE can be generalized to k-SUBGRAPH ISOMORPHISM for the case when the pattern graph F has treewidth at most t. Towards this we need a result analogous to Lemma 5.7 for trees, which can be proved using the separation properties of graphs of treewidth at most t. This will lead to an algorithm with running time $2.851^k \cdot n^{\mathcal{O}(t)}$.

5.5   Other Applications

Marx [33] gave algorithms for several problems based on matroid optimization. The main theorem in his work is Theorem 1.1 [33] on which most applications of [33] are based. The proof of the theorem uses an algorithm to find representative sets as a black box. Applying our algorithm (Theorem 1 of this paper) instead gives an improved version of Theorem 1.1 of [33].

Proposition 5.4. *Let $M = (E, \mathcal{I})$ be a linear matroid where the ground set is partitioned into blocks of size ℓ. Given a linear representation A_M of M, it can be determined in $\mathcal{O}(2^{\omega k \ell} ||A_M||^{\mathcal{O}(1)})$ randomized time whether there is an independent set that is the union of k blocks. ($||A_M||$ denotes the length of A_M in the input.)*

Finally, we mention another application from [33] which we believe could be useful to obtain single exponential time parameterized and exact algorithms.

ℓ-MATROID INTERSECTION   **Parameter:** k
Input: Let $M_1 = (E, \mathcal{I}_1), \ldots, M_1 = (E, \mathcal{I}_\ell)$ be matroids on the same ground set E given
  by their representations $A_{M_1}, \ldots, A_{M_\ell}$ over the same field $\mathbb{F}$ and a positive integer k.
Question: Does there exist k element set that is independent in each M_i ($X \in \mathcal{I}_1 \cap \ldots \cap \mathcal{I}_\ell$)?

Using Theorem 1.1 of [33], Marx [33] gave a randomized algorithm for ℓ-MATROID INTERSECTION. By using Proposition 5.4 instead we get the following result.

Proposition 5.5. *ℓ-MATROID INTERSECTION can be solved in $\mathcal{O}(2^{\omega k \ell} ||A_M||^{\mathcal{O}(1)})$ randomized time.*

6 Conclusion

In this paper, we gave fast algorithms computing representative families of independent sets in linear matroids. Moreover, we demonstrate that for families of uniform matroids even better algorithms are available. We also show interesting links between representative families of matroids and the design of single-exponential parameterized and exact exponential algorithms. We believe that these connections have a potential for a wide range of applications.

The natural questions left open are the following.

- What is the right time to compute a minimal q-representative family for a family of independent sets of a linear matroid? Can it be computed in time linear in the size of the minimal q-representative family or superlinear lower bounds can be found?

- It would be interesting to find faster algorithms even for special classes of linear matroids.

- Finally, the only matroids we used in our algorithmic applications were graphic, uniform, and partition matroids. It would be interesting to see what kind of applications can be handled by other type of matroids.

References

[1] N. ALON, R. YUSTER, AND U. ZWICK, *Color-coding*, J. Assoc. Comput. Mach., 42 (1995), pp. 844–856.

[2] O. AMINI, F. V. FOMIN, AND S. SAURABH, *Counting subgraphs via homomorphisms*, SIAM J. Discrete Math., 26 (2012), pp. 695–717.

[3] J. BANG-JENSEN AND G. GUTIN, *Digraphs*, Springer Monographs in Mathematics, Springer-Verlag London Ltd., London, second ed., 2009. Theory, algorithms and applications.

[4] A. BJÖRKLUND, T. HUSFELDT, P. KASKI, AND M. KOIVISTO, *Narrow sieves for parameterized paths and packings*, CoRR, abs/1007.1161 (2010).

[5] A. BJÖRKLUND, T. HUSFELDT, AND S. KHANNA, *Approximating longest directed paths and cycles*, in Proceedings of the 31st International Colloquium, Automata, Languages and Programming (ICALP 2004), vol. 3142 of Lecture Notes in Comput. Sci., Springer, 2004, pp. 222–233.

[6] H. L. BODLAENDER, *On linear time minor tests with depth-first search*, J. Algorithms, 14 (1993), pp. 1–23.

[7] H. L. BODLAENDER, M. CYGAN, S. KRATSCH, AND J. NEDERLOF, *Solving weighted and counting variants of connectivity problems parameterized by treewidth deterministically in single exponential time*, CoRR, abs/1211.1505 (2012).

[8] B. BOLLOBÁS, *On generalized graphs*, Acta Math. Acad. Sci. Hungar, 16 (1965), pp. 447–452.

[9] J. BUNCH AND J. HOPCROFT, *Triangular factorization and inversion by fast matrix multiplication*, Mathematics of Computation, 28 (1974), pp. 231–236.

[10] J. CHEN, J. KNEIS, S. LU, D. MÖLLE, S. RICHTER, P. ROSSMANITH, S.-H. SZE, AND F. ZHANG, *Randomized divide-and-conquer: improved path, matching, and packing algorithms*, SIAM J. Comput., 38 (2009), pp. 2526–2547.

[11] J. CHEN, S. LU, S.-H. SZE, AND F. ZHANG, *Improved algorithms for path, matching, and packing problems*, in Proceedings of the18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), SIAM, 2007, pp. 298–307.

[12] N. COHEN, F. V. FOMIN, G. GUTIN, E. J. KIM, S. SAURABH, AND A. YEO, *Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem*, J. Comput. System Sci., 76 (2010), pp. 650–662.

[13] T. H. CORMEN, C. LEISERSON, R. RIVEST, AND C. STEIN, *Introduction to Algorithms*, The MIT Press, Cambridge, Mass., second ed., 2001.

[14] M. CYGAN, J. NEDERLOF, M. PILIPCZUK, M. PILIPCZUK, J. M. M. VAN ROOIJ, AND J. O. WOJTASZCZYK, *Solving connectivity problems parameterized by treewidth in single exponential time*, in Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011), IEEE, 2011.

[15] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized complexity*, Springer-Verlag, New York, 1999.

[16] J. EDMONDS, *Optimum branchings*, J. Res. Nat. Bur. Standards Sect. B, 71B (1967), pp. 233–240.

[17] F. V. FOMIN AND D. KRATSCH, *Exact exponential algorithms*, Springer, 2011.

[18] F. V. FOMIN, D. LOKSHTANOV, V. RAMAN, S. SAURABH, AND B. V. R. RAO, *Faster algorithms for finding and counting subgraphs*, J. Comput. System Sci., 78 (2012), pp. 698–706.

[19] P. FRANKL, *An extremal problem for two families of sets*, European J. Combin., 3 (1982), pp. 125–127.

[20] H. N. GABOW AND S. NIE, *Finding a long directed cycle*, ACM Transactions on Algorithms, 4 (2008).

[21] H. T. HSU, *An algorithm for finding a minimal equivalent graph of a digraph*, J. Assoc. Comput. Mach., 22 (1975), pp. 11–16.

[22] R. IMPAGLIAZZO, R. PATURI, AND F. ZANE, *Which problems have strongly exponential complexity*, Journal of Computer and System Sciences, 63 (2001), pp. 512–530.

[23] S. JUKNA, *Extremal combinatorics*, Springer Verlag Berlin Heidelberg, 2011.

[24] T. KLOKS, *Treewidth, Computations and Approximations*, vol. 842 of Lecture Notes in Computer Science, Springer, 1994.

[25] J. KNEIS, D. MÖLLE, S. RICHTER, AND P. ROSSMANITH, *Divide-and-color*, in Proceedings of the 34th International Workshop Graph-Theoretic Concepts in Computer Science (WG 2008), vol. 4271 of Lecture Notes in Computer Science, Springer, 2008, pp. 58–67.

[26] I. KOUTIS, *Faster algebraic algorithms for path and packing problems*, in Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008), vol. 5125 of Lecture Notes in Computer Science, 2008, pp. 575–586.

[27] I. KOUTIS AND R. WILLIAMS, *Limits and applications of group algebras for parameterized problems*, in Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2009), vol. 5555 of Lecture Notes in Computer Sci., Springer, 2009, pp. 653–664.

[28] S. KRATSCH AND M. WAHLSTRÖM, *Representative sets and irrelevant vertices: New tools for kernelization*, in Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS 2012), IEEE, 2012, pp. 450–459.

[29] L. LOVÁSZ, *Flats in matroids and geometric graphs.*, in In Combinatorial surveys (Proc. Sixth British Combinatorial Conf., Royal Holloway Coll., Egham), Academic Press, London, 1977, pp. 45–86.

[30] S. MARTELLO, *An algorithm for finding a minimal equivalent graph of a strongly connected digraph*, Computing, 21 (1978/79), pp. 183–194.

[31] S. MARTELLO AND P. TOTH, *Finding a minimum equivalent graph of a digraph*, Networks, 12 (1982), pp. 89–100.

[32] D. MARX, *Parameterized coloring problems on chordal graphs*, Theor. Comput. Sci., 351 (2006), pp. 407–424.

[33] ——, *A parameterized view on matroid optimization problems*, Theor. Comput. Sci., 410 (2009), pp. 4471–4479.

[34] M. MITZENMACHER AND E. UPFAL, *Probability and computing: Randomized algorithms and probabilistic analysis*, Cambridge University Press, 2005.

[35] B. MONIEN, *How to find long paths efficiently*, in Analysis and design of algorithms for combinatorial problems (Udine, 1982), vol. 109 of North-Holland Math. Stud., North-Holland, Amsterdam, 1985, pp. 239–254.

[36] D. M. MOYLES AND G. L. THOMPSON, *An algorithm for finding a minimum equivalent graph of a digraph*, J. ACM, 16 (1969), pp. 455–460.

[37] K. MUROTA, *Matrices and matroids for systems analysis*, vol. 20, Springer, 2000.

[38] M. NAOR, L. J. SCHULMAN, AND A. SRINIVASAN, *Splitters and near-optimal derandomization*, in Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS 1995), IEEE, 1995, pp. 182–191.

[39] J. G. OXLEY, *Matroid theory*, vol. 3, Oxford University Press, 2006.

[40] C. H. PAPADIMITRIOU AND M. YANNAKAKIS, *On limited nondeterminism and the complexity of the V-C dimension*, J. Comput. Syst. Sci., 53 (1996), pp. 161–170.

[41] J. PLEHN AND B. VOIGT, *Finding minimally weighted subgraphs*, in Proceedings of the 16th Workshop on Graph-Theoretic Concepts in Computer Science (WG 1991), vol. 484 of Lecture Notes in Comput. Sci., Springer, 1991, pp. 18–29.

[42] Z. TUZA, *Applications of the set-pair method in extremal hypergraph theory*, in Extremal problems for finite sets (Visegrád, 1991), vol. 3 of Bolyai Soc. Math. Stud., János Bolyai Math. Soc., Budapest, 1994, pp. 479–514.

[43] ——, *Applications of the set-pair method in extremal problems. II*, in Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993), vol. 2 of Bolyai Soc. Math. Stud., János Bolyai Math. Soc., Budapest, 1996, pp. 459–490.

[44] R. WILLIAMS, *Finding paths of length k in $O^*(2^k)$ time*, Inf. Process. Lett., 109 (2009), pp. 315–318.

[45] V. V. WILLIAMS, *Multiplying matrices faster than Coppersmith-Winograd*, in Proceedings of the 44th Symposium on Theory of Computing Conference (STOC 2012), ACM, 2012, pp. 887–898.